uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería Informática
Curso 2017-2018

Trabajo Fin de Grado

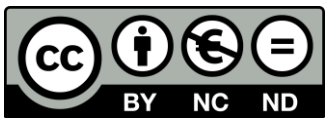
"Protección de Credenciales en Trabajos Fin de Grado: Análisis y elaboración de materiales didácticos"

José Manuel González Osona

Tutora

Ana Isabel González-Tablas Ferreres

Leganés, 25 de septiembre de 2018



RESUMEN

El campo de la ingeniería informática sigue evolucionando rápidamente y la seguridad informática va tomando fuerza con el paso del tiempo. Pero es un hecho que los conocimientos del usuario normal no mantienen el paso de los avances y en algunos casos ni los propios estudiantes del campo. Hoy en día, la mayor parte de los sistemas informáticos requieren de una autentificación del usuario frente al sistema para poder acceder a los servicios ofrecidos. El problema es que existen varios ataques para usurpar las credenciales de usuario y generalmente no se han implantado mecanismos de seguridad para protegerlas. Esta deficiencia es notoria hasta en los alumnos del grado de ingeniería informática de la UC3M, por lo que se realizará un estudio para analizar las posibles causas de estas deficiencias.

El análisis en cuestión se centrará en los TFG de alumnos de la UC3M para comprobar si se implementan mecanismos de protección de credenciales de usuario. Previo al análisis se realizará una propuesta de soluciones que tendrá en cuenta las técnicas actualmente recomendadas y el grado de dificultad de implementación para que los alumnos no deban dedicar un tiempo excesivo a la adición de las mismas. El análisis de los TFG tiene una etapa previa en la que se filtrarán aquellos TFG que posean tratamiento de datos de usuario y sistemas de autentificación. En base a las muestras significativas se realiza un análisis individual de cada una de ellas y una extracción de estadísticas para proponer distintas causas por las que no se lleva a cabo la implementación de los mecanismos de seguridad en los TFG.

Tomando como base las causas de la no implementación de los mecanismos de seguridad en los TFG y las propuestas previamente realizadas, se generarán un conjunto de materiales didácticos para ayudar a concienciar y ofrecer distintas implementaciones de los mecanismos de seguridad a los alumnos acompañados de una prueba de validación para revisar que se han afianzado los conceptos trabajados.

Palabras clave: funciones hash, criptografía, materiales didácticos, análisis de TFG, seguridad informática.

DEDICATORIA

A mi tutora Ana por permitirme la oportunidad de llevar a cabo este TFG que me ha ayudado a profundizar mis conocimientos en seguridad informática.

A mis amigos Daniel, Dark y Laura que, aunque no conozcan conceptos de informática me ayudaron en todo lo posible para seguir adelante y revisar el documento.

A mis amigos de la universidad del grupo UC3M Pals, Diego, Edu, Andrés, Hervás y Manuel que han sido siempre una inspiración para superarme a mí mismo y he disfrutado cuatro años que han sido increíbles.

Por último y no menos importante a mi familia más cercana que son mi abuela Esperanza con su fe ciega en mí, y a mis padres Esperanza y Manuel que siempre me han impulsado ser la mejor versión de mi mismo y me han aguantado en todas las malas circunstancias, que son muchas.

ÍNDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1. INTRODUCCIÓN

En este capítulo se hace una descripción global del trabajo de fin de grado. Primeramente, se expondrá la motivación del trabajo, en la que se expone una visión global de la situación respecto a las credenciales de usuario. Después se citarán los objetivos que tiene que cumplir el TFG y, por último, se explica la estructura del documento para aclarar qué se tratará en cada capítulo.

1.1. Motivación del Trabajo

En la actualidad la seguridad informática va cobrando cada vez más fuerza, y poco a poco se observa una mayor concienciación de la gente por la seguridad de sus datos privados. Aunque esto es cierto, en el propio campo de la ingeniería informática la seguridad no parece ser demasiado bien hallada entre los profesionales de la rama, esto se debe a la implicación de una mayor complejidad resultante a la hora de diseñar un nuevo sistema informático, ya que se debe asegurar que también sea seguro.

Existe una mala concepción de lo que es la seguridad informática. Generalmente, cuando se habla de seguridad informática se piensa en el cifrado de los datos con sistemas de cifrado simétricos o asimétricos. Si bien esto es cierto, no es verdad que la solución a todos los problemas de seguridad consista en añadir la capa de cifrado a los datos para evitar que sean leídos de forma no autorizada. Existe un equilibrio que se debe mantener entre la seguridad, la funcionalidad y la usabilidad del sistema informático.

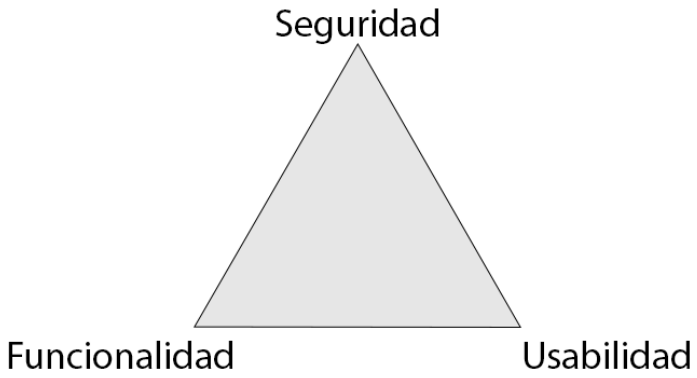


Figura 1: Triángulo de la seguridad

En algunos casos los sistemas informáticos requieren que el usuario sea capaz de identificarse para poder ofrecer las funcionalidades correctas. El método a través del cual se suelen identificar los usuarios frente a un sistema son las credenciales de usuario.

Las credenciales de usuario son el medio por el cual un usuario es capaz de identificarse ante un sistema informático. Éstas son generalmente conocidas como el *nombre de usuario* y *contraseña* de una cuenta. Adicionalmente, las credenciales de usuario pueden pertenecer a tres categorías:

1. **Algo que el usuario sabe.** El ejemplo más común es conocer una contraseña, PIN, o número que se espera que sea secreto para el resto del mundo salvo para el usuario y el sistema.

2. **Algo que el usuario tiene.** Puede ser un token, o una "tarjeta inteligente". En este caso un ejemplo sencillo sería el DNIe que posee un circuito con certificados PKI para estar reconocidos como ciudadanos españoles, avalado por la Policía Nacional (debido al PKI). Otro posible ejemplo es el Pasaporte que posee información personal como nombres, apellidos, fecha de nacimiento y también guarda certificados para autentificarse con las entidades policiales correspondientes.
3. **Algo que el usuario es.** Generalmente se refiere a datos biométricos que son, datos personales obtenidos a partir de un tratamiento técnico específico, relativos a las características físicas, fisiológicas o conductuales de una persona física que permitan o confirmen la identificación única de dicha persona, como imágenes faciales o datos dactiloscópicos [1].

El problema de las credenciales de usuario yace en que el usuario común, en caso de tener varias cuentas con diversos sistemas informáticos, suele emplear usuarios y contraseñas similares para evitar tener que recordar varias combinaciones de usuario-contraseña.

También es cierto que hay usuarios que saben que no deberían tener la misma contraseña para varias plataformas, pero la forma de escoger la contraseña suele ser bastante débil o simple, como por ejemplo el cumpleaños de alguien conocido o el propio, el nombre de una mascota, el nombre de un familiar, etc.

Además, está el caso en el que se le obliga al usuario a seguir una política de contraseña segura que generalmente especifica el número mínimo de caracteres, una letra mayúscula, una minúscula y un número. El problema está en que varios usuarios escogerán su contraseña común y la adaptarán para cumplir con los requisitos mínimos de la política de contraseña segura. Por ejemplo, si la contraseña es *jmgouc3m* un usuario podría simplemente añadir la primera letra como mayúscula y con esa modificación ya se cumple la política. Lo que ocurre es que estas pequeñas modificaciones son menores y cualquier atacante puede deducirlas e incorporarlas a las reglas para los ataques por diccionario. Por tanto, un usuario normal no tendrá unas credenciales de usuario realmente seguras y posiblemente sean las mismas para distintas cuentas.

Haciendo referencia a lo mencionado anteriormente, uno de los ataques más comunes para obtener las credenciales de usuario de la gente es *"credential stuffing"*, que en líneas generales se basa emplear combinaciones de usuario y contraseñas que han sido obtenidas después de un ataque a una base de datos u obtenidas en una *"password dump site"* para intentar encontrar una combinación correcta de usuario-contraseña y acceder a los datos privados del usuario.

Teniendo en cuenta todo esto, es imprescindible proteger la confidencialidad de las credenciales de usuario que se empleen en los sistemas informáticos.

1.2. Objetivos

Los objetivos de este TFG son, primeramente, concienciar a los alumnos de la UC3M sobre la importancia de proteger las credenciales de usuario en las aplicaciones o sistemas que desarrollen, ya no específicamente para sus TFG, sino para futuros sistemas en los que participen. Aun así, la audiencia de este TFG no está limitada a los alumnos de la UC3M, sino a cualquier otro estudiante de ingeniería informática que no esté excesivamente relacionado o sea conocedor de las prácticas de seguridad informática.

Además, se explicarán varios conceptos de seguridad que son básicos para comprender el posterior análisis de los TFG de años previos. En dicho análisis, se explicará brevemente las posibles vulnerabilidades observadas del sistema/aplicación respecto a las credenciales de usuario. En el caso de que existan dichas vulnerabilidades y no hayan sido tratadas, se expondrán soluciones para estas. Posteriormente, se realizarán estadísticas con los datos obtenidos del análisis de cada TFG, en los que se intentará reflejar, entre otros, el nivel de concienciación respecto a la seguridad y el nivel de implicación ya sea al menos mencionando la seguridad o llevando a cabo la implementación de un mecanismo de seguridad.

Por último, se generarán unos materiales didácticos que servirán a los lectores como ejercicios guiados para aprender sobre algunas de las prácticas en la protección de las credenciales de usuario previamente explicadas.

1.3. Estructura del documento

El presente documento está dividido en distintas secciones. Cada una de estas secciones abarca una parte del trabajo desarrollado en el proyecto. Los capítulos en los que se encuentra dividido el documento son:

1. **Introducción:** en este capítulo se ofrece una visión global del proyecto en la que se explican los motivos y objetivos a lograr.
2. **Estado de la cuestión:** esta sección recoge el estado actual respecto a la protección de las credenciales de usuario. Se comentarán conceptos básicos de seguridad informática y se explicarán las técnicas recomendadas para asegurar la confidencialidad de las credenciales de usuario. Por último, se propondrán qué soluciones se deberían llevar a cabo teniendo en cuenta los futuros TFG de los alumnos de la UC3M.
3. **Análisis de TFG recientes:** en esta parte se explicará el proceso de selección y filtrado de los TFG para su posterior análisis respecto a la inclusión o no de la protección de las credenciales de usuario. En base a las muestras recogidas se obtendrán una serie de estadísticas para comprender cual ha sido la tendencia a lo largo del tiempo respecto a la inclusión de la confidencialidad de las credenciales de usuario.
4. **Diseño de materiales didácticos:** en este apartado se comenta el proceso de elección y diseño de los materiales didácticos, se expondrán también la serie de ejercicios guiados que permitirán a los lectores comprender mejor los conceptos explicados para que en un futuro puedan aplicarse. Por último, se provee de una prueba de validación para que los lectores puedan confirmar que han conseguido afianzar los conceptos tratados en los materiales didácticos.
5. **Gestión del proyecto:** en este ámbito se expondrá la planificación del proyecto, incluyendo las horas realizadas y la distribución de estas a lo largo de los meses.
6. **Entorno socioeconómico y marco regulador:** en esta parte se realiza un estudio del entorno socioeconómico y un estudio del marco regulador que puede afectar al proyecto.
7. **Conclusiones:** en este capítulo se concretará si los objetivos del TFG han sido cumplidos y se expondrán posibles trabajos futuros.
8. **Bibliografía:** en este apartado aparecerán todos los enlaces y referencias bibliográficas que se han empleado para el desarrollo del TFG.
9. **Resumen en inglés:** en este capítulo se incluye un resumen del proyecto en inglés.

2. ESTADO DE LA CUESTIÓN

En este capítulo se explican las técnicas actuales de protección de credenciales de usuario y conceptos de seguridad informática para poder comprender los mecanismos empleados con mayor facilidad. Posteriormente, se exponen las distintas opciones que pueden emplear los alumnos o lectores para que el sistema en el que estén trabajando posea unas credenciales de usuario confidenciales.

2.1. Situación Actual

Previa a la descripción de la situación actual sobre los métodos empleados o recomendados para la protección de las credenciales de usuario se introducirán a continuación unos conceptos técnicos de la seguridad informática. Estos conceptos son clave para comprender el porqué de los mecanismos de seguridad empleados y facilitar una mayor comprensión de los puntos que se discuten a lo largo del documento.

El primero de los conceptos es la **vulnerabilidad**. Una vulnerabilidad es una brecha de seguridad en un sistema informático o una debilidad en la seguridad en el sistema o aplicación. Generalmente se debe a fallos en el diseño o un error en la implementación que puede permitir a un atacante causar daños a los activos o usuarios. Los **activos** en este ámbito se refieren a los datos o a la aplicación o sistema. En cuanto al término **usuarios**, se refiere a los propietarios y a los usuarios finales que emplean el servicio ofrecido. Éstos últimos pueden ser usuarios finales que usan la aplicación u otras entidades o empresas que emplean el servicio ofrecido.

El tipo de daño causado a los activos es distinto del que puede realizar a los usuarios. En el caso de los activos un posible daño sería el borrado o pérdida de datos, incluida también la modificación de estos. Respecto a los usuarios se podría pensar en ataques que no permiten a los usuarios acceder al servicio como pueden ser los ataques *DoS* (*Denial of Service* o Ataques de denegación de servicio) o el uso de los datos personales de los usuarios al haber atacado a los activos, entre los que generalmente se encuentran cuentas, contraseñas o información personal como el nombre, apellidos, edad, residencia, etc. Los ataques *DoS* como su nombre indica se centran en evitar que el sistema o aplicación pueda proveer el servicio que ofrece, generalmente se consigue a través de un gran número de peticiones al sistema de forma que éste gaste tiempo de cómputo en atender esas peticiones y no la de un usuario real o a través de la generación de objetos en el sistema para llenar la memoria del sistema y ralentizar o bloquear su funcionamiento.

El siguiente concepto es el de **amenaza**. Una amenaza es la intención de realizar un ataque por parte de una entidad hacia una de las vulnerabilidades del sistema o aplicación. Existen las **amenazas conocidas** que suelen estar recopiladas en un repositorio, pero también existen las **amenazas desconocidas** que son aquellas que se identifican estudiando el funcionamiento del sistema o aplicación durante un periodo de tiempo. Algunas de las amenazas relacionadas con la protección de credenciales pueden ser un *Key Logger* o *sniffers*. Los *key logger* pueden ser un software o hardware que se emplea para registrar las pulsaciones del teclado y después enviarlas a algún otro sitio conocido por el atacante para poder revisar los datos sobre qué cosas se han escrito. Los *sniffers* por otro lado son programas que se centran en la monitorización y análisis de los paquetes de la red.

El último de los conceptos a introducir es el de **ataque**. Un ataque es la acción de llevar a cabo técnica para explotar las vulnerabilidades del sistema o aplicación, es decir, lo que un atacante

haría para aprovechar una vulnerabilidad. Algunos de los ejemplos más comunes son, un ataque de fuerza bruta, *DNS poisoning*, ataques *DoS* y *DDoS* (*Distributed Denial of Service*). El ataque *DNS poisoning* se centra en la introducción de datos corruptos en el servidor *DNS (Domain Name Server)* que conlleva a la redirección de los usuarios a páginas web maliciosas del atacante.

Una vez se han explicado los conceptos de la seguridad informática que ayudan a la comprensión de la situación y algunos de los ataques realizados, se comentarán a continuación los mecanismos que actualmente se recomienda seguir para proteger o mantener un nivel de seguridad aceptable a la hora de hablar de las credenciales de usuario.

La primera de las practicas consiste en no limitar el conjunto de caracteres para las contraseñas e intentar tener contraseñas largas. Actualmente varias empresas o servicios reducen el conjunto de caracteres que se pueden emplear a la hora de escoger la contraseña y la longitud de estas para evitar posibles ataques de inyección SQL y XSS (*Cross-Site Scripting*). Estas prácticas, aunque puedan llegar a evitar posibles ataques de los mencionados previamente, pueden simplificar el trabajo de otros y más comunes como el de un ataque de fuerza bruta debido a la reducción del conjunto de caracteres y longitud de las contraseñas. Cabe destacar que longitudes de contraseñas excesivamente grandes pueden permitir ataques *DoS*.

Un ataque de fuerza bruta es un ataque que consiste principalmente en un atacante que configura un conjunto de valores predeterminados, luego realiza peticiones a un servidor empleando los valores previamente dispuestos y realiza un posterior análisis de los resultados obtenidos de las peticiones. Los atacantes pueden emplear un ataque de fuerza bruta puro, que es un escenario en el que se prueban todas las combinaciones posibles o un ataque por diccionario. Un ataque por diccionario es uno en el que se crea un conjunto predefinido de palabras a probar.

En el caso de contraseñas un ataque de diccionario haría uso de contraseñas obtenidas de bases de datos previamente atacadas, contraseñas más usadas y palabras derivadas de las anteriores. La diferencia entre estos dos ataques es que en el primero al probar todas las posibles combinaciones debe emplear una mayor cantidad de los recursos para realizar el ataque. En el caso del ataque por diccionario se limita el conjunto de valores a emplear por lo que debería ser más rápido, al menos en tiempo. Hay que mencionar que la capacidad de éxito de los ataques es distinta, el primero conseguirá eventualmente la combinación correcta, aunque el tiempo empleado sea excesivo y no factible para un ataque, sin embargo, el éxito del ataque por diccionario depende de si el diccionario posee la contraseña entre sus valores predefinidos. Existen herramientas para realizar estos ataques como:

- Bruter [2]
- THC Hydra [3]
- John the Ripper [4]
- Brutus [5]

Adicionalmente, algunas de las contraseñas más usadas o expuestas se pueden encontrar aquí [6].

Otra técnica recomendada es el uso de hashes en vez del almacenamiento en claro de las contraseñas en la base de datos. El uso de las funciones hash permite guardar un valor que es único para la contraseña en cuestión debido a las propiedades de las funciones hash. Si se emplea este método a la hora de la autentificación se puede comprobar el valor del hash

almacenado con el valor del hash calculado sobre la credencial ingresada, de esta forma si la credencial introducida es correcta ésta coincidirá con el hash almacenado. Siguiendo este esquema en caso de que la base de datos o el sistema se vea comprometido, sufra un ataque y se filtren datos de las credenciales de los usuarios, la información que será recabada sería una lista de hashes y no la propia contraseña en claro. Esto evitaría ataques a los usuarios en otras plataformas, ya que, como se comentó previamente los usuarios suelen mantener las mismas contraseñas y usuarios para distintas plataformas. Para comprender en mayor profundidad cómo es posible, a continuación, se dará una explicación formal de qué son las funciones hash, cuáles son sus propiedades y beneficios.

Una función hash es una función matemática al que, dado un input o mensaje, independientemente de su longitud, genera un output que es único y de tamaño fijo, generalmente llamado hash, hash *digest*, *digest* o *hashed value*. Las funciones hash deben cumplir con las siguientes propiedades para ser consideradas seguras ante ataques:

- **Resistencia a pre-imágenes:** dado un valor de hash *h* debería ser difícil encontrar otro mensaje o input *m* de forma que *h = hash(m)*. Esta propiedad hace referencia a la cualidad de una función de una sola dirección, de forma que teniendo solamente el hash resultante es imposible generar el input del que procede.
- **Resistencia a segunda pre-imagen:** dado un input m_1, debería ser difícil encontrar otro input distinto m_2 de tal forma que hash(m_1) = hash(m_2). En el caso de las credenciales de usuario, se refiere a que, si se obtiene el hash de la contraseña, un input incorrecto que no sea la contraseña no debe generar el mismo hash.
- **Resistencia a colisiones:** debería ser suficientemente difícil encontrar dos mensajes m_1 y m_2, siendo $m_1 \neq m_2$, tal que hash(m_1) = hash(m_2). Este par, en caso de encontrarse, se llama una colisión.

La resistencia a colisiones de una función implica la resistencia a segunda pre-imagen, pero no asegura resistencia a pre-imágenes. La diferencia entre la resistencia a segunda pre-imagen y resistencia a colisiones radica en la elección de m_1. En el caso de la resistencia a segunda pre-imagen el atacante tiene un valor m_1 fijo y debe encontrar otro m_2 que tenga mismo hash, el atacante en este caso no puede escoger m_1. En caso de resistencia a colisiones el atacante tiene la libertad de escoger m_1 y m_2.

Además de estas propiedades mencionadas existen otras propiedades que las funciones de hash deben cumplir. Estas propiedades son las siguientes:

- Debe ser determinista, el mismo input deber generar el mismo resultado siempre.
- Debe ser rápido en computar el valor de cualquier mensaje dado.
- No debe ser factible generar el mensaje dado su valor de hash, excepto que se prueben todos los mensajes posibles que idealmente son infinitos.
- Un pequeño cambio en el mensaje o input inicial debe cambiar el hash resultante tanto de forma que no parezca relacionado con el hash previo. Es decir, dado un m_1 con su respectivo h_1, si se modifica m_1 obteniendo m_2 siendo h_2 su hash, h_2 no debe parecer correlacionado con h_1.
- No debe ser factible encontrar dos mensajes diferentes con el mismo valor de hash resultante. Es decir, dado m_1 y m_2 se debe cumplir que hash(m_1) ≠ hash(m_2).

Tal y como se ha visto, las funciones hash tienen la capacidad de dado un input generar un valor de hash único, pero éstas son susceptibles a las colisiones. Por ello es importante saber que el espacio de valores para una función hash es:

$$|H| = 2^n$$

Donde n es el número de bits de salida de la función hash.

Una vez se ha comprendido el funcionamiento de una función hash, lo siguiente es destacar que no se debe usar cualquier función de hash para evitar el robo de credenciales en claro. Las funciones hash cumplen otros objetivos como asegurar la integridad de archivos.

Una de las funciones hash más empleadas hasta el momento para uso general era MD5. Es una función hash que tiene 128 bits de salida. El problema es que se ha demostrado que MD5 no es resistente a las colisiones. Por tanto, el algoritmo no es que esté roto como tal, pero al ser susceptible a ataques de colisión no es una opción viable a la hora de emplearlo para almacenar los valores de hash de las contraseñas, ya que el ataque por colisiones puede hallar otro input que tenga el mismo valor hash. Varios de los ataques que ha sufrido MD5 aparecen en el siguiente artículo [7]. Este algoritmo actualmente se suele emplear para asegurar la integridad de archivos. Adicionalmente, otra de las funciones más empleadas es SHA1 (*Secure Hash Algorithm 1*) a pesar de la existencia de sus nuevas versiones SHA2 y SHA3. La función SHA1 no es recomendada para el uso como función hash según el NIST (*National Institute of Standards and Technology*) [8] según su política de funciones [9]. Otro motivo para evitar el uso de SHA1 es el reciente ataque el 23 de febrero de 2017 que obtuvo una colisión, dicho ataque es *SHAttered* [10].

Por tanto, para el propósito de no almacenar las credenciales en claro se recomienda emplear las siguientes funciones:

- Argon2
- PBKDF2
- *scrypt*
- *bcrypt*

El uso de funciones hash de por si no es suficiente para asegurar un nivel de seguridad aceptable para proteger a las credenciales de usuario. Se recomienda el uso de las funciones hash que permitan añadir un *salt* para así añadir más aleatoriedad y evitar que ataques como los ataques de diccionario y las *rainbow tables* sean factibles, ataques que se explicarán más adelante.

El término *salt* en el ámbito de la criptografía se refiere a datos aleatorios, compréndase aleatorio como pseudoaleatorio ya que no existe ningún método que permita ofrecer datos puramente aleatorios. Este *salt* es empleado en las funciones hash o funciones de una dirección que genera un hash *digest*. Los *salt* están relacionados directamente con los *nonce* que son números generados de forma pseudoaleatoria o aleatoria. El problema de los *salt* recae evidentemente en que sean lo suficientemente aleatorios para que el resultado de hashear el *salt* y la contraseña sea único. La forma de obtener estos valores de *salt* criptográficamente fuertes es a través de los CSPRNG o *Cryptographically Secure Pseudo-Random Number Generator*. Los CSPRNG deben cumplir dos propiedades que permiten asegurar que son criptográficamente seguros y son las siguientes:

- **Debe satisfacer la prueba del siguiente bit.** La prueba consiste en que dados los primeros k bits de una secuencia aleatoria, no hay ningún algoritmo de tiempo polinómico que pueda predecir el (k+1)-ésimo bit con una probabilidad de éxito mayor a un 50%.
- **Debe soportar extensiones de compromiso de estado.** Esto significa que en caso de que parte o toda la información del estado haya sido revelado (o predicho correctamente), debe de ser imposible reconstruir el flujo de números aleatorios generados con anterioridad a la obtención del estado.

A continuación, se muestra una tabla con algunos CSPRNG para distintos lenguajes:

TABLA 1: CSPRNG PARA DISTINTOS LENGUAJES

Lenguaje	CSPRNG
PHP	Mcrypt_create_iv, openssl_random_pseudo_bytes
Java	Java-security.SecureRandom
.NET	System.Security.Cryptography.RNGCryptoSerivceProvider
Ruby	SecureRandom
Python	os.urandom
Perl	Math::Random::Secure
C/C++ de la API de Windows	CryptGenRandom
GNU/Linux o Unix	Lecturas de /dev/random, /dev/urandom o /dev/arandom

Otros aspectos que se deben tener en cuenta cuando se empleen *salts* es que nunca se debe reusar un *salt*. Cada *salt* debe ser usado una única vez como parámetro para hashear una contraseña y solamente con un usuario. Adicionalmente, el uso de un mismo *salt* para más de un usuario puede resultar en que si dos usuarios emplean la misma contraseña, el hash obtenido y almacenado en la base de datos sea el mismo. Esto significa que si una de las dos cuentas con el mismo hash se ven comprometidas el atacante obtendrá acceso a las dos cuentas.

Además, en caso de que el sistema o aplicación permita la modificación de las contraseñas de los usuarios se debería volver a generar un nuevo *salt* para la nueva contraseña en vez de volver a reutilizar el *salt* de la contraseña anterior. Respecto a la longitud del *salt* empleado, ésta dependerá del máximo permitido por la función hash, pero generalmente es recomendable usar *salts* largos a *salts* cortos. Como recomendación si no se sabe qué longitud emplear, se puede emplear la longitud de salida de la función hash que se va a emplear. Si se emplea un SHA512 la salida es de 512 bits (64 bytes) asique el *salt* debería de tener una longitud de 64 bytes al menos.

En cuanto al almacenamiento de los *salts* para su posterior uso, como por ejemplo validar los inicios de sesión de un usuario, pueden almacenarse en claro o cifrados. La decisión recae en el tiempo que tarde el cifrado empleado, ya que se debe recordar que a mayor seguridad menor usabilidad por parte del usuario y el tiempo de respuesta para un simple inicio de sesión puede ser excesivo.

Tal y como se mencionó el uso del *salt* en funciones hash sirve para evitar una serie de ataques que de otra forma serían posibles. Dichos ataques son *lookup tables*, *reverse lookup tables* y *rainbow tables* los cuales se explican a continuación.

Las *lookup tables* son un conjunto de contraseñas asociadas a su hash correspondiente. Estas tablas son rápidas en tiempo, pero pesadas en memoria, por eso las *lookup tables* se preparan y pre computan antes para llegar a poder ser capaces de obtener la contraseña de varios hashes, siempre y cuando la contraseña se encuentre entre el conjunto de inputs escogidos. Es otro método de hacer un ataque de fuerza bruta, tan solo que esta vez se obtienen los valores primero, luego se almacenan y se hace consultas en vez de que para cada input se genere el hash correspondiente. Es un intercambio de memoria y tiempo de cómputo.

Otra versión de las *lookup table* son las *reverse lookup table* que se basan al igual que las *lookup table* en pre computar un conjunto de valores a pares valor-hash. La diferencia principal entre las dos técnicas es el método de búsqueda, en el caso de las *reverse lookup table* se escoge el hash del que se quiere obtener el valor inicial y la búsqueda en la tabla se hace buscando ese hash, en vez de ir buscando fila a fila de forma secuencial en la tabla.

Las *rainbow tables* son unas tablas que intentan un modelo en el que ni el espacio ni el tiempo de cómputo sea excesivo. Hasta ahora el método para obtener las contraseñas era generar un conjunto de inputs y hacer el hash correspondiente e ir probando. Él objetivo en sí no cambia, pero si el cómo se realiza. A la hora de usar estas tablas existen dos funciones para que llevarlo a cabo, una es la **función hash** y otra una **función reduce**. La función hash será, como su nombre indica, la que recibirá el input del dominio del texto (que son las contraseñas) a el dominio del hash que se espera sea único.

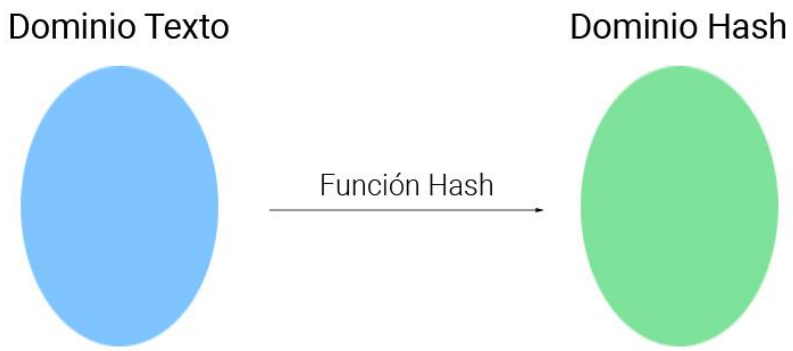


Figura 2: Dominio texto y dominio hash (función hash)

La función reduce lo que aporta es un cambio de dominio desde el dominio de los hashes hasta el dominio del texto. Esto no quiere decir que la función reduce sea la función inversa de la función hash, ya que esto es imposible, sino que es una función que al darle un hash de input genera un output perteneciente al dominio del texto. Es necesario aclarar que el resultado de aplicar la función reduce a un hash no devuelve el valor del texto previo a ser hasheado, sino que devuelve otra palabra del espacio del texto.

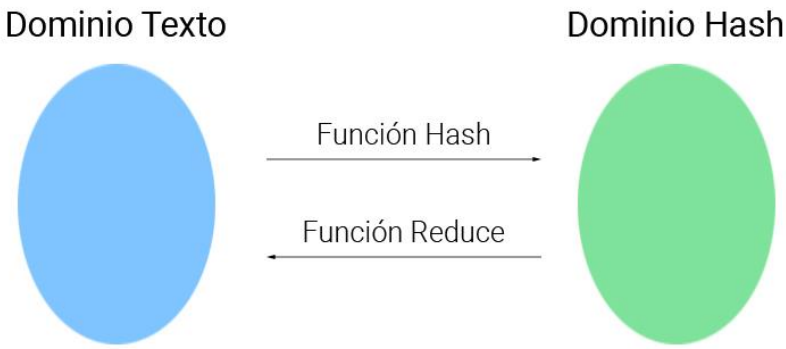


Figura 3: Dominio texto y dominio hash (función hash y reduce)

Una vez que se tienen las dos funciones lo que se hace es encadenar el uso de las funciones, una al resultado de la otra. De forma que, inicialmente se tiene un input, del dominio del texto, al que se le aplica la función hash. A dicho hash se le aplica la función reduce y se obtiene un output del dominio del texto. Este procedimiento sigue tantas iteraciones como se especifique dando lugar a las cadenas.

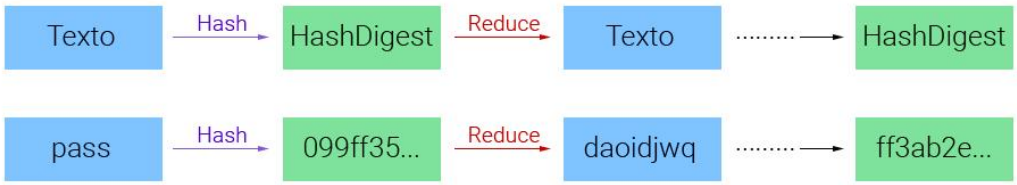


Figura 4: Cadena de una rainbow table

A la hora de almacenar los resultados, no se almacenan todos los resultados intermedios, sino que se emplea el valor de inicio de la cadena y el valor del hash final de la cadena. Entonces, realmente lo que se almacena son pares de palabra de inicio de una cadena con el hash final obtenido tras la ejecución de las iteraciones especificadas. De esta forma se reduce la cantidad de datos a guardar.

Una vez se han generado las cadenas se puede empezar a usar el algoritmo para hallar el input de un hash. El algoritmo seguiría los siguientes pasos:

1. Comprobar si el hash está en los hashes finales de cada cadena, si está ahí salir del bucle.
2. Si no está entre los hashes finales, reducir el hash y volver a hashear el resultado.
3. Nueva Iteración
4. Si el hash coincide con alguno de los hashes finales, la cadena con la que coincide contiene el texto plano que generó ese hash.

Para recuperar el valor lo restante es empezar en la cadena con la que coincidió el hash y con el valor inicial almacenado, iterar tantas veces hasta que se obtenga el hash inicial que se estaba comprobando y el paso anterior será el valor del dominio del texto que está hasheado.

Las *rainbow table* aun así tienen algunos problemas. Cuando se lleva a cabo la creación de una *rainbow table* es posible que se den colisiones, es decir, que dos valores distintos del dominio del texto den un mismo valor del dominio de hashes. El problema que esto ocasiona es que puede derivar en que dos cadenas que tienen un comienzo distinto tengan el mismo hash final, y compartan los valores desde que ocurre la colisión hasta que se termine la cadena.

Otro caso es que se formen bucles cuando se está creando una cadena. Esto se debe a que la función *reduce* pueda devolver un valor que ha sido anteriormente hasheado en las iteraciones previas de la cadena.

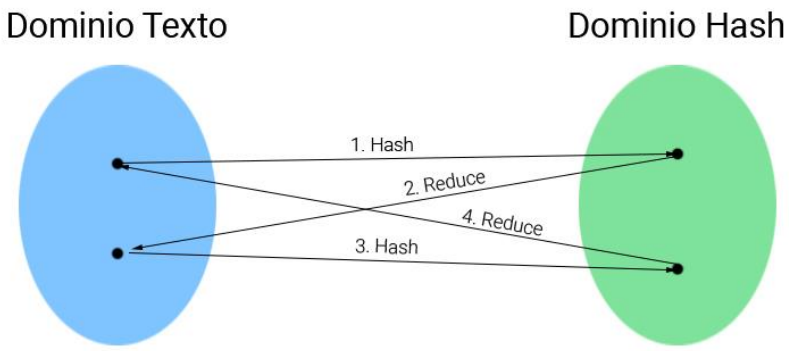


Figura 5: Bucle entre los dominios texto y hash

Debido a estos problemas con las colisiones y los bucles, no existe la certeza que un valor hash de un texto se transforme a través de la función reduce a otro texto distinto. A pesar de la potencia que tienen las *rainbow tables* para generar hashes y textos, éstas no aseguran que dentro de las cadenas generadas se vaya a hallar el hash deseado. Lo único que pueden hacer es aumentar la probabilidad de que se encuentre el hash buscado lo más próxima a 1.

La forma de solucionar las colisiones, por ende, las uniones de distintas cadenas y los bucles era finalizar la cadena en un punto determinado, específicamente en un hash que cumpliese ciertas propiedades o tras un número definido de ejecuciones. En el caso que dos cadenas con inicio distinto acabaran en el mismo valor significaría que ha ocurrido una colisión en algún punto de las cadenas y por tanto una de ellas se descartaría. En el caso de los bucles, estos se detectaban si una cadena excedía un tiempo de ejecución hasta llegar al punto determinado, en estos casos había una gran probabilidad que existiera un bucle y se descartaba la cadena entera.

Todo esto se hacía en el primer acercamiento a las *rainbow table,* más tarde se presentó un artículo [11] que propuso un nuevo sistema para el cálculo de las cadenas. Hasta entonces lo que se hacía era crear pequeñas tablas con un conjunto de mensajes iniciales m_i a los que se les aplicaba una única función de reducción. El nuevo acercamiento propone hacer una única tabla, pero en este caso emplear varias funciones de reducción en cada columna. Cada columna es el resultado de aplicar la función de reducción a un hash y por tanto una iteración.

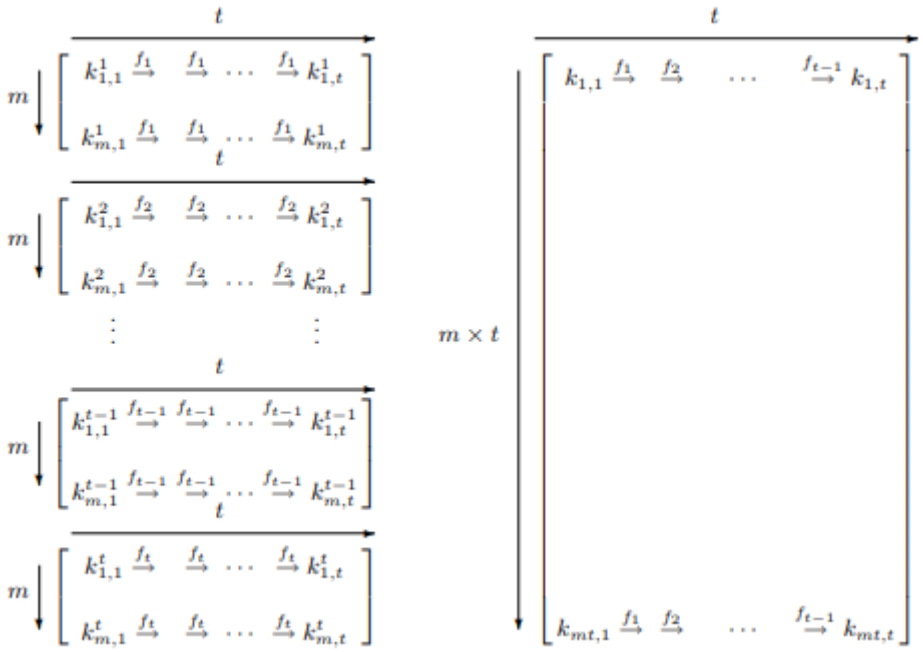


Figura 6: Classic tables (izquierda) vs Raibow table (derecha). Fig2 en el artículo [11].

En base a este nuevo método de cálculo de las cadenas, la cantidad de unión de cadenas se vio reducida, ya que, el único momento en el que dos cadenas se deberían unir sería en el caso en que se obtuviera el mismo valor en la misma columna. Si se daba ese caso significaba que a partir de esa columna ambas cadenas iban a realizar las mismas operaciones y debido a esa colisión se debían de unir ambas cadenas. Esto no significa que las colisiones dejaran de ocurrir, solo que al ocurrir en distintas columnas la función de reducción siguiente es distinta y las cadenas no convergerían. En el caso de una cadena de longitud *t*, lo cual significa que hay *t* distintas funciones de reducción, la probabilidad de que haya una unión es de $\frac{1}{t}$. Por último, hay que mencionar que no pueden aparecer bucles en estas nuevas *rainbow table* debido a que en cada columna se emplea una distinta función *reduce*.

La siguiente técnica se emplea para generar claves lo suficientemente seguras a partir de contraseñas de los usuarios. Previa a la explicación se deben aclarar los conceptos de clave y contraseña. Una contraseña es un conjunto de caracteres generados por un usuario que deben ser secretos, generalmente empleados para identificarse en un sistema. Una clave son los datos que se emplean para habilitar y deshabilitar funciones criptográficas o para verificar la autentificación de los usuarios. Dicho de otra forma, la clave en este caso serán los datos que usará el sistema para verificar si el usuario es quien dice ser comparando el valor almacenado y el valor generado en base a la contraseña introducida por el usuario.

Hasta ahora, se ha visto el uso de las funciones hash y la aplicación de estas con *salts* criptográficos. La secuencia de cambios sobre el tratamiento de las credenciales de usuario abarca desde que simplemente se almacenan las contraseñas en claro hasta la aplicación de una función hash para evitar que si el sistema es comprometido todas las contraseñas almacenadas sean públicas. Luego, con la aparición de distintos ataques sobre las funciones hash se vio la necesidad de ampliar la seguridad afectando a la aleatoriedad del input empleado en las funciones hash, esto se realizó a través de los *salts* generados por los CSPRNG.

El siguiente paso es el uso de técnicas de *key stretching,* las cuales se centran en convertir una clave débil, en este caso la contraseña del usuario, a una clave más segura frente a ataques. El

ataque por excelencia es el de fuerza bruta y las técnicas de *key stretching* aumentan los recursos necesarios para la generación de cada clave y por ende dificulta el ataque de fuerza bruta y de diccionario. Con el simple hecho de aplicar estas técnicas, los ataques de diccionarios se ven forzados a ser más complejos y dinámicos, lo que suele generar una mayor necesidad en la capacidad de cómputo. El ataque queda reducido a probar todas las posibles combinaciones, que es la esencia del ataque, pero increíblemente costosa y generalmente no factible. La otra opción para llevar a cabo el ataque es generar posibles claves a partir de la modificación de una posible contraseña inicial. Cabe destacar que las técnicas de *key stretching* no aportan un método de prevenir esta segunda opción de ataque, pero sí que afectan al atacante de forma que se aumenta los recursos que debe emplear para cada intento que quiera realizar. Las técnicas de *key stretching* son generalmente el uso de una función hash criptográficamente segura varias veces mediante un bucle. En caso de no usar una función hash se puede emplear el cifrado en bloque.

Las funciones KDF o *key derivation function* son funciones que derivan claves a partir de un valor secreto inicial como puede ser una clave maestra o contraseña. Las KDF se suelen emplear como una técnica de *key stretching* de forma que se crean claves más largas. Un ejemplo de cómo funcionaría una KDF sería el siguiente:

$$ClaveDerivada = KDF(contraseña, salt, número\ de\ iteraciones)$$

Como se puede apreciar la clave derivada final, que es más resistente a ataques de fuerza bruta, se genera a partir de la KDF a la que se le dan como argumentos en este caso:

- **Contraseña:** es la contraseña secreta que conoce el usuario
- **Salt:** es el *salt* criptográfico generado por un CSPRNG
- **Número de iteraciones:** es el número de veces que se aplica la función o subfunción para llevar a cabo la técnica de *key stretching.*

De esta forma, lo que se consigue es que el sistema asignará este nuevo valor como clave del usuario en vez de usar la contraseña que es presumiblemente débil y predecible. Este valor se puede recuperar para realizar las autenticaciones ya que muchas de las KDF emplean funciones hash, las cuales tienen la propiedad de ser deterministas y frente a un mismo input generan el mismo resultado.

Es necesario aclarar que las KDFs no tienen como objetivo único el derivar una única clave para identificar a los usuarios y proteger sus credenciales. Las KDFs pueden cumplir el propósito de generar varias claves derivadas a partir de una clave maestra para cifrar distintas sesiones con distintos usuarios o la expansión de una clave para hacerla más larga. Debido a esto las funciones que se emplean con el objetivo de generar claves para proteger las credenciales de usuario y ser capaces de realizar una verificación son:

- bcrypt
- scrypt
- PBKDF2
- Argon2

De entre todas estas Argon2 es la más nueva y la más recomendada para usar en la seguridad de las credenciales de usuario. Argon2 es el algoritmo ganador de la PHC (*Password Hashing*

Competition) que empezó en otoño del 2012 y acabó en Julio de 2015 cuando se anunció al ganador [12].

Otra posibilidad es el uso de las HMAC (*Hashed-based Message Authentication Code*) para la protección de las credenciales de usuario. Las HMAC son un tipo específico de MAC (*Message Authentication Code*) generado a partir de una función hash junto con una clave secreta. Las funciones HMAC pueden verificar simultáneamente la autentificación y la integridad del mensaje. Las HMAC son al fin y al cabo las funciones hash a las que se les incorpora el uso de una clave secreta para aportar la autentificación. Cabe destacar que la seguridad aportada por la HMAC depende del algoritmo o función hash empleada para dicha HMAC. La estructura de las HMAC está definida en el RFC 2104 [13]. En la sección 2 del RFC 2014 se explica la composición de la HMAC y los parámetros empleados. Para una compresión más sencilla se emplearán las mismas variables descritas en el RFC 2104 [13].

La función hash sobre la que se basa la HMAC se denota como H. La clave secreta empleada será K. Se asume que la función H opera sobre bloques de datos de tamaño B, con tamaño $B = 64\ bytes$ para las funciones SHA-1, MD5 y RIPEMD-128/160. La longitud de salida de la función H en bytes se denota L, siendo $L = 16$ para MD5 y $L=20$ para SHA-1. La longitud de la clave K puede ser de una longitud máxima de B, aquellas HMAC que tengan una K mayor de B aplicarán el hash primero a K y usarán el L resultante como K para el HMAC. Además, existen dos variables más que son $ipad = el\ byte\ 0x36\ repetido\ B\ veces$ y $opad = el\ byte\ 0x5C\ repetido\ B\ veces$. La fórmula de las HMAC es finalmente:

$$HMAC = H(K\ XOR\ opad\ ||\ (H(K\ XOR\ ipad\ ||\ texto)))$$

Cabe destacar que en caso de que K sea de longitud inferior a B, se rellenará con bytes de cero (0x00) por la derecha hasta que se llegue al tamaño de B.

En el caso de utilizar la opción de las HMAC hay que tener unos conceptos claros que son:

- Usar una sola clave para el sistema.
- Proteger esta clave como cualquier otra clave privada.
- Almacenar la clave fuera de donde se almacenen las credenciales.
- Generar una clave con un método que genere datos pseudoaleatorios criptográficamente fuertes, por ejemplo, un CSPRNG o KDF.

Por último, remarcar la importancia del almacenamiento seguro de la clave empleada para las HMAC, ya que, la ventaja de estas frente a las funciones hash dependen sola y únicamente de que la clave secreta siga siéndolo.

2.2. Propuesta de soluciones

En este apartado se explicará qué medidas se deben llevar a cabo para poder asegurar que las credenciales de los usuarios del sistema o aplicación que se esté desarrollando estén seguras.

Esta serie de pautas a seguir están seriamente ligadas a los conceptos mencionados en el punto de la situación actual. Esto se debe a que se conocen cuáles deberían ser las pautas, pero sólo dentro del círculo de las personas que se dedican a la criptografía o les atrae debido a que es un campo complejo y difícil de iniciarse en él. También, hay que tener en cuenta que las propuestas de soluciones a llevar a cabo están orientadas a alumnos que desarrollan un TFG. Por ello, las soluciones propuestas tienen el objetivo de añadir un nivel de seguridad aceptable para los

sistemas desarrollados sin tener que generar un trabajo excesivamente pesado para los alumnos.

Política de contraseña

Lo primero que se debería hacer es no limitar los caracteres que puede emplear un usuario para escoger su contraseña. Si bien es cierto que no se debe limitar ni el conjunto de caracteres ni la longitud de la contraseña, para evitar ataques DoS es recomendable fijar un límite, pero no debe ser muy restrictivo. Un posible máximo para la contraseña serían 160 caracteres. El hecho de no limitar el conjunto de caracteres para la contraseña pone en riesgo que suframos ataques *XSS* (*Cross Site Scripting*) y de inyección SQL, por tanto, es recomendable emplear *whitelists* y *blacklists* para evitar el uso de caracteres no deseados o patrones que sean maliciosos como puede ser <*script*></*script*>.

También es recomendable seguir una política de creación de contraseñas para que el usuario añada cierto nivel de complejidad a la contraseña. Una de las políticas más comunes es que al menos contenga una letra mayúscula, al menos contenga una letra minúscula y al menos contenga un dígito. Estas pautas aumentan el número de combinaciones necesarias para los ataques de fuerza bruta, aunque es cierto que pueden dar pistas a los ataques por diccionario, por ello, se debería avisar al usuario que no modifique de forma evidente su contraseña para cumplir los requisitos mínimos como puede ser cambiar la primera letra a mayúscula y añadir al final un digito o conjunto de dígitos relacionados con sus datos personales. Lo que si es necesario es la especificación de la longitud mínima de la contraseña que según el NIST debe ser de 8 caracteres, especificado en el apartado 5.1.1.1 de la guía de identificación [14].

Funciones hash

Una vez el usuario tiene libertad para escoger su contraseña es necesario saber qué métodos emplear para poder almacenar de forma segura las credenciales de usuario. Para ello se deberían usar funciones hash sobre las contraseñas y almacenar el valor hash obtenido. De esta forma la autenticación de los usuarios puede continuar y en caso de que la base de datos donde residen las contraseñas se vea expuesta a un ataque, los datos almacenados sobre las contraseñas no les serán inmediatamente útiles a los atacantes. Sin embargo, el hecho de usar una función hash no es suficiente, ya que, se deben usar funciones hash que sean consideradas criptográficamente seguras y actualizadas. Algunas estas funciones son:

- bcrypt
- scrypt
- PBKDF2
- Argon2
- SHA2
- SHA3

Es necesario ser consciente que las funciones hash no simplemente cumplen el objetivo del almacenamiento seguro de contraseñas, sino que fueron diseñadas para otros propósitos, por eso se debe saber qué función se está empleando y si fue diseñada para el propósito en el que se va a emplear. Adicionalmente, algunas de las funciones hash mencionadas están diseñadas para ralentizar la obtención de los hashes, algo que debe tenerse en cuenta cuando se vaya a emplear en un sistema o aplicación en la que la velocidad de respuesta sea crucial.

Salt criptográfico

Aunque se empleen funciones hash criptográficamente seguras aún se puede añadir un grado mayor de protección a los usuarios mediante el uso de *salts* criptográficos. El motivo reside en que varios usuarios emplearán contraseñas similares, ya usadas por otra gente o fácilmente deducibles. Por lo tanto, si simplemente se aplica la función hash, si los atacantes se percatan que dos hashes son idénticos solamente puede ser porque ambos usuarios poseen la misma contraseña y solo tendrían que conseguir la contraseña de uno de los dos usuarios para acceder a cualquiera de las dos cuentas. Para evitar este tipo de situaciones los *salts* criptográficos son un conjunto de caracteres pseudo-aleatorios generados por CSPRNGs que se suelen añadir al inicio o al final de las contraseñas de los usuarios antes de hashear la contraseña. De esta forma lo que se hashea es una concatenación del *salt* y la contraseña, y aunque dos usuarios empleen la misma contraseña como los *salts* serán distintos el hash resultante será también distinto.

En cuanto al uso de los *salts* hay que ser bastante cauteloso porque no se debe de reutilizar un valor de *salt* nunca. No se debe usar para más de un usuario ni tampoco para el caso que un usuario quiera cambiar su contraseña actual. Cada vez que la contraseña cambie o se quiera hashear una nueva contraseña se debe emplear un valor nuevo y distinto de *salt* para cada una de ellas. A la hora de almacenar el hash se debe almacenar también el *salt* con el que se usó la función hash de forma que se pueda realizar el proceso de autentificación cada vez que el usuario quiera volver a iniciar sesión. El *salt* en este caso puede ser almacenado en claro porque será distinto para cada usuario y de una longitud al menos igual al tamaño de salida de la función hash que se usa.

Cifrado de datos

Otra opción es el cifrado de los datos que se creen que son críticos. La decisión de emplear un cifrado de datos debe sopesarse y decidir si realmente es necesario o no. Para poder tomar una decisión correcta y argumentada se debe saber para qué se emplea cada tipo de cifrado sabiendo cuáles son sus ventajas y desventajas. Los tipos de cifrado dependiendo del tipo de clave empleada son:

El cifrado simétrico, que se basa en el uso de una clave secreta para el mundo salvo para el emisor y receptor del mensaje. El mensaje se cifra con la clave secreta y se envía a través del canal al receptor. El receptor empleando la clave secreta previamente compartida o acordada, descifra el mensaje. La misma clave sirve para el cifrado y descifrado, por eso debe ser secreta. Algunos ejemplos de sistemas de cifrado simétricos son DES, Triple DES (TDES o 3DES) y AES.

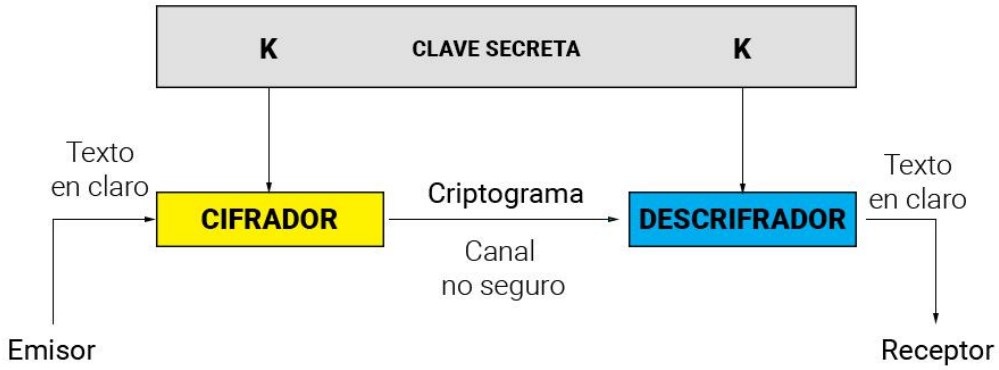


Figura 7: Esquema de cifrado simétrico

Las ventajas de los sistemas simétricos son:

- Rapidez, el cifrado simétrico es mucho más rápido que el asimétrico.
- Uso menor de los recursos del sistema en comparación con el cifrado asimétrico.
- Necesidad de una única clave para cifrar y descifrar datos.

Las ventajas mencionadas son solamente aquellas que tienen relación con el uso de un sistema de cifrado simétrico para el cifrado de datos de por ejemplo una base de datos. Los sistemas de cifrado simétrico tienen otras ventajas adicionales cuando son empleados para el intercambio de mensajes.

La desventaja principal en este caso es que al usar una única clave simétrica para el cifrado y descifrado, si esta se ve comprometida todos los datos dejarían de ser confidenciales.

El cifrado asimétrico, también conocido como de clave pública, se basa el en uso de dos claves. Una clave se llama clave pública y como su nombre indica es pública y conocida por todo el mundo. La otra clave se llama clave privada, esta clave debe ser secreta y no revelarse a nadie. Un ejemplo de un sistema de cifrado asimétrico popular es RSA.

En el caso del cifrado asimétrico la forma en la que se cifra cambia respecto a los sistemas simétricos. Supóngase que Alicia quiere enviar un mensaje a Benito, ambos poseen su par de claves público y privada. En el caso que Alicia quiera enviarle un mensaje cifrado a Benito, el mensaje deberá cifrarse con la clave pública de Benito. Una vez Benito reciba el mensaje podrá descifrarlo si usa su clave privada. A continuación, se muestra un esquema para aclarar el funcionamiento.

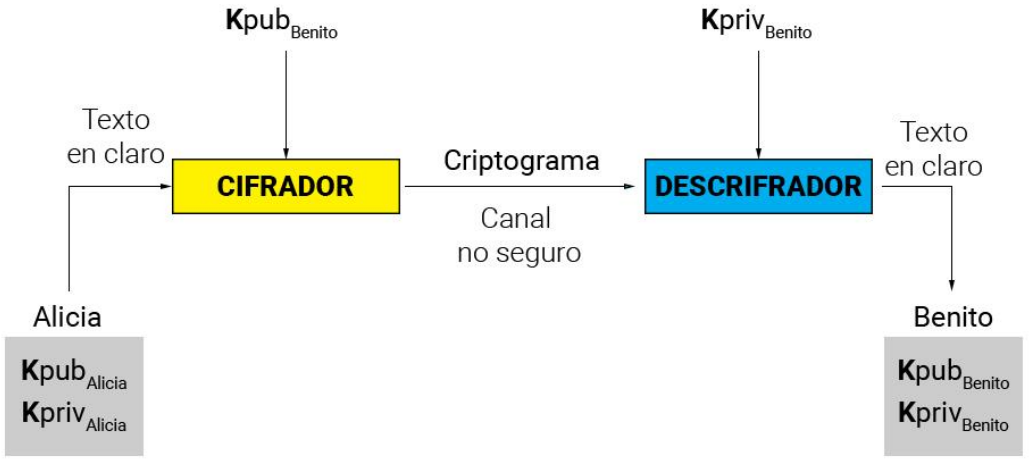


Figura 8: Esquema de cifrado asimétrico

La ventaja principal de los sistemas asimétricos es que si se emplean claves largas el sistema de cifrado asimétrico es más pesado y por ende más difícil de romper. Por el contrario, las desventajas son:

- Lento en comparación con el cifrado simétrico
- Usa más recursos del sistema que el cifrado simétrico
- La pérdida de la clave privada resulta en la imposibilidad de descifrar los datos, ya que, las claves públicas y privadas no pueden ser derivadas una a partir de la otra.

El cifrado híbrido es un cifrado que emplea los dos tipos de cifrado previos para aprovechar al máximo sus cualidades. El cifrado híbrido se centra en el uso del cifrado simétrico para el cifrado de los datos, de forma que el proceso es rápido y además eficaz ya que se asegura la confidencialidad de los datos. El cifrado asimétrico se emplea en este caso para proteger la clave de cifrado del cifrado simétrico. De esta forma se emplea el cifrado asimétrico que es más pesado, pero más potente para proteger la clave simétrica y el cifrado de los datos se realiza de forma simétrica para acelerar el proceso. Éste es un método bastante útil para cuando se quiere asegurar la confidencialidad de los datos, pero un cifrado simétrico se queda corto y un cifrado asimétrico aporta un exceso de tiempo y cómputo para los datos que se quieren cifrar.

Adicionalmente, se puede diferenciar dependiendo del modo de cifrado empleado. Los modos que se muestran a continuación son los recomendados por el NIST en su documento de recomendación para modos de cifrado por bloques [15]. Los modos recomendados son los siguientes:

- ECB, *Electronic Code Book Mode*
- CBC, *Cipher Block Chaining Mode*
- CFB, *Cipher Feedback Mode*
- OFB, *Output Feedback Mode*
- CTR, *Counter Mode*

El modo ECB (*Electronic Code Book*) es un modo de cifrado que, en base a una clave, asigna un bloque cifrado a cada uno de los textos en plano. Es decir, que para cada bloque M_i le corresponde su bloque cifrado C_i empleando la misma clave K para todos los bloques, de modo que C*i* = Cifrado (M*i*, K). Cada uno de los bloques se cifra y descifra de forma independiente. Las ventajas que aporta este modo de cifrado es que permite paralelizar el cifrado y descifrado. Permite que se puedan cifrar bloques sin necesidad de cifrar los bloques anteriores y, por ende,

los errores de transmisión no se propagan entre bloques. Algunas de las desventajas es que bloques repetidos dan como resultado el mismo bloque cifrado.

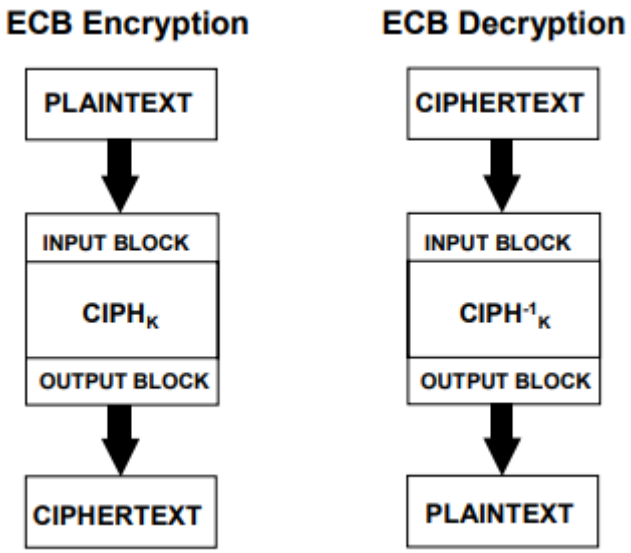


Figura 9: Modo de operación ECB. Figure 1 del articulo [15]

El siguiente modo de cifrado es CBC (*Cipher Block Chaining*) que es un modo en el que se enlazan los textos en plano con el bloque previo cifrado. Este modo necesita de un vector de inicialización para combinarlo con el primer bloque de texto plano. El cifrado de un bloque C_i sigue la siguiente fórmula C_i = Cifrado (C_{i-1} XOR M_i, K) donde C_0 = vector de inicialización. El proceso de descifrado sigue la siguiente fórmula M_i = Descifrar (C_i, K) XOR C_{i-1}. Este modo de operación no permite la paralelización en el proceso de cifrado debido a que el resultado de la operación del cifrado actual depende de la iteración anterior, por el contrario, el proceso de descifrado puede ser paralelizado porque todos los bloques son accesibles.

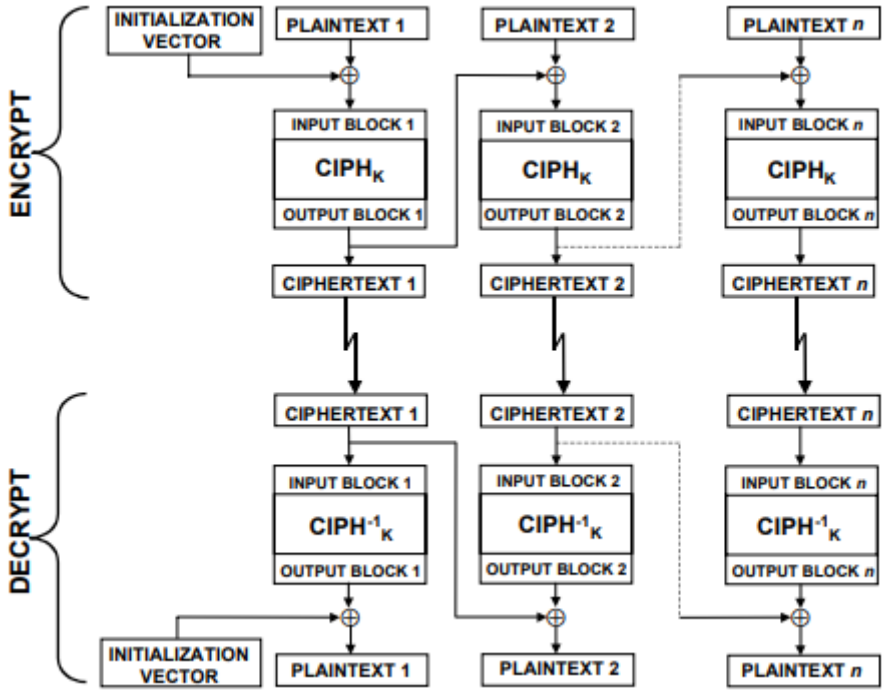


Figura 10:Modo de operación CBC. Figure 2 del artículo [15]

Otro de los modos de cifrado es el CFB (*Cipher Feedback*) que emplea una realimentación de m bits con un m arbitrario. Los m bits más significativos de la salida del cifrador se extraen y se operan mediante un XOR con m bits del texto en claro, el resultado son m bits del criptograma que se envían a la línea de transmisión y también realimentan el registro de desplazamiento.

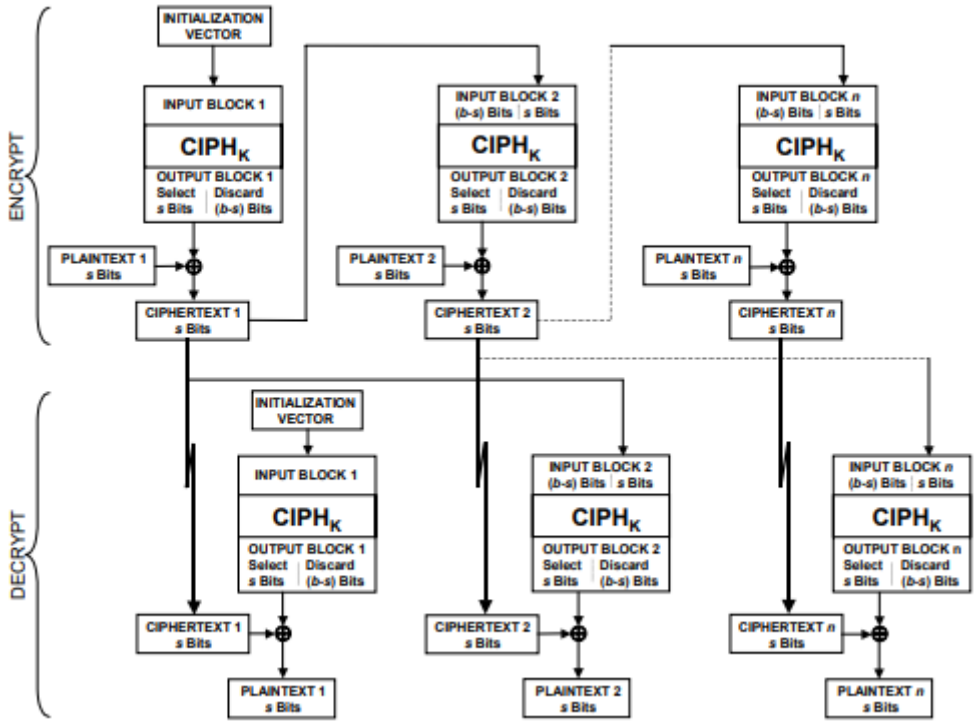


Figura 11: Modo de operación CFB. Figure 3 del artículo [15]

El modo de cifrado siguiente es OFB (*Output Feedback*) en el que el cifrador obtiene el bloque cifrado con que aplica un XOR al texto en claro para generar el cifrado del texto en plano, este modo de cifrado requiere de un vector de inicialización. La realimentación de este modo de cifrado se obtiene de la salida del propio cifrador. Este modo de cifrado permite la conversión de un cifrador de bloque en uno de flujo ya que la serie cifrante no depende del texto en claro y se opera sobre bloques no sobre segmentos.

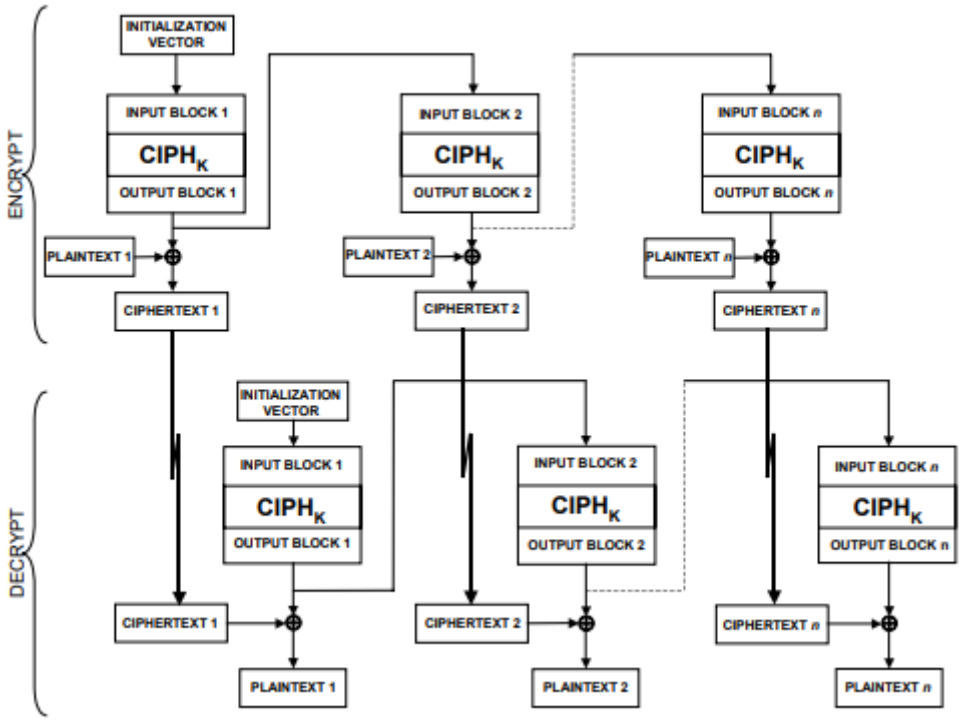


Figura 12: Modo de operación OFB. Figure 4 del artículo [15]

El último modo de cifrado es CTR (*Counter*) en el que se generan claves independientemente del contenido de los bloques cifrados. En este modo de cifrado, se suman números *nonce* (*number used once*) y se le suma un valor de un contador. A partir del valor resultante y la clave, se obtiene el bloque con el que se aplicará la operación XOR con el bloque de texto en plano y así obteniendo el bloque cifrado. El tamaño del contador es del tamaño del bloque, y al igual que OFB permite la conversión de un cifrador de bloque en uno de flujo. Adicionalmente este modo permite la paralelización debido al acceso aleatorio.

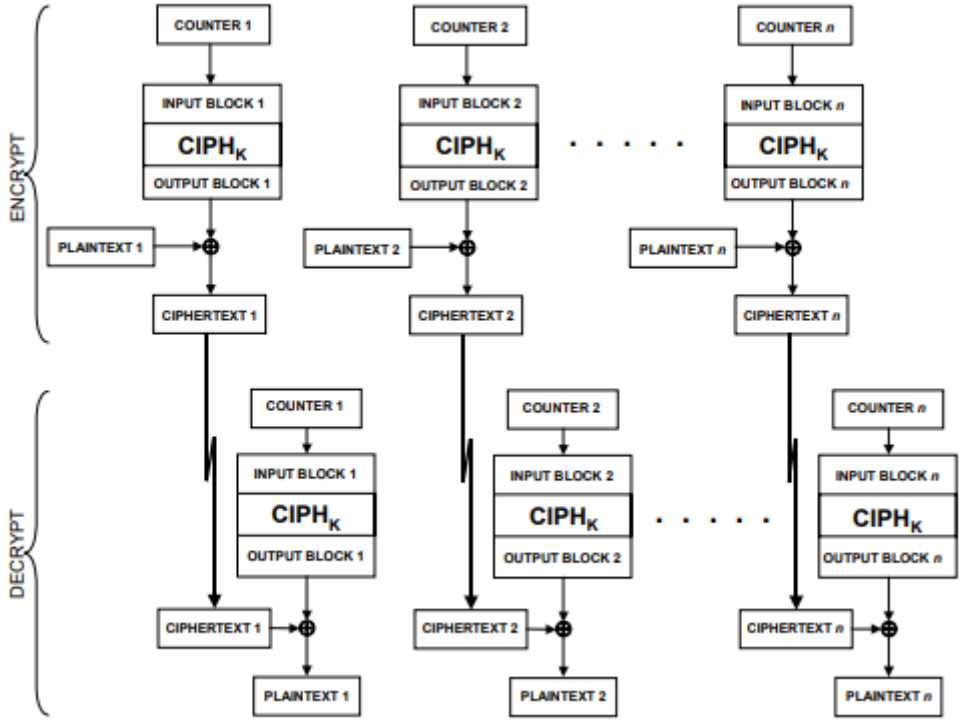


Figura 13: Modo de operación CTR. Figure 5 del artículo [15]

El objetivo de la explicación de estos modos de cifrado es saber cómo se van a cifrar los datos en caso de que en alguna implementación de un algoritmo de cifrado se deba especificar el modo de cifrado a emplear.

Existen otra serie de acciones que se pueden llevar a cabo para asegurar la seguridad de las credenciales de usuario sin necesariamente trabajar con ellas. Una de ellas es limitar las responsabilidades de acceso a la base de datos. Esto se debe a que muchas personas emplean por mayor facilidad el usuario administrador de la base de datos como usuario para realizar consultas que no necesitan que lo haga el administrador. Para estos casos es una buena idea crear otros usuarios para que el acceso a la información dentro de la base de datos sea limitado.

Otra acción que se puede llevar a cabo es la limitación de permisos a archivos y carpetas en sistemas Unix. En caso de que se tenga un sistema Unix se debería prestar especial atención a los permisos que tiene cada usuario sobre el sistema. En este caso se recomienda aplicar la política de prohibir todos los accesos y otorgar sólo los permisos exclusivamente necesarios contra la política de prohibir o denegar los permisos que no debería tener el usuario. La ventaja de la primera política es que se asegura que el usuario tiene sola y exactamente los permisos básicos necesarios para operar en el sistema. La segunda política no es tan recomendable ya que depende de la capacidad para denegar permisos a todo aquello que no se quiera dar acceso y puede haber errores o que no se prohíban permisos que deberían estar denegados.

Por último, hay que mencionar que a pesar de que el uso de las HMAC es bastante recomendado como solución para asegurar la confidencialidad de las credenciales de usuario, debido a la fuerte necesidad de mantener la clave empleada segura y secreta para que sean mejor opción que las funciones hash análogas no se recomienda su uso. Ya que esto implicaría que los autores tendrían que pensar en un modo seguro de almacenamiento de la clave para la función HMAC y un modo seguro de la transmisión de la clave lo que al fin y al cabo terminaría complicando el diseño del sistema frente al uso de una de las funciones hash previamente expuestas.

3. ANÁLISIS DE TFG RECIENTES

3.1. Análisis de Trabajos de Fin de Grado

En este apartado se realizará un análisis individual de cada uno de los TFG que se han tomado como muestras válidas para el estudio de la protección de las credenciales de usuario en los TFG de alumnos previos de la UC3M [16].

El proceso que se ha llevado a cabo ha sido dividido en dos etapas, en la primera etapa se hace una revisión superficial de los trabajos en la que se filtraban muestras dependiendo de si tenía relación con las credenciales de usuario o desarrollo de una aplicación, evidentemente las muestras significativas que sirven para este estudio son aquellas que hagan uso de usuarios registrados en el sistema y usen un sistema de autentificación. En la segunda etapa, una vez se han filtrado todos los trabajos estudiados hasta el momento que son del orden de 400 registros obtenidos del buscador e-Archivo UC3M [17], se procede a hacer un estudio en mayor profundidad estudiando que datos se almacenan sobre los usuarios y si emplean credenciales de usuario, así como un conocimiento general del sistema que se ha desarrollado.

Tal y como se ha mencionado, se ha usado el buscador de e-Archivo para obtener los distintos TFG, para ello, los parámetros de búsqueda han sido especificar la comunidad de trabajos académicos. Después, se han listado los trabajos académicos por área de conocimiento, que en este caso es la Informática. Finalmente se han ordenado por fecha de publicación en orden descendente y se van obteniendo registros que corresponden con cada uno de los TFG almacenados en el repositorio de e-Archivo UC3M. Respecto al filtrado de los TFG, se ha tenido en cuenta también la existencia de tres ramas de especialización en el Grado en Ingeniería Informática que son Computadores, Computación y Sistemas de Información. Teniendo en cuenta la existencia de estas tres ramas, es bastante probable que los trabajos relacionados con la rama de Computadores y Sistemas de Información traten el desarrollo de aplicaciones, así como los de la rama de Computación están centrados en el desarrollo de algoritmos y técnicas de Inteligencia Artificial para generar *bots* o agentes que cumplan un rol en un juego. Adicionalmente, mientras se realizaba el estudio se ha tenido en cuenta también el Doble Grado en Ingeniería Informática y Administración de Empresas, debido a que generalmente llevaban a cabo el desarrollo de un sistema con fines comerciales, de ayuda al comercio o de ayuda a la toma de decisiones de otras empresas.

Una vez se ha realizado el fin de la segunda etapa de evaluación de los TFG se obtuvieron un total de 25 muestras significativas que como máximo datan de cuatro años previos al estudio realizado.

Para el análisis individual de cada trabajo escogido, se ha tenido en cuenta que aplicación se ha desarrollado y si emplea usuarios. Hay que mencionar que entre las muestras obtenidas se han añadido algunas que no requieren del registro del usuario porque principalmente se sospecha que esa decisión de diseño les ahorraría tiempo y preocupaciones por la seguridad que se debería añadir al sistema. El esquema general que se ha seguido para cada análisis individual es una breve introducción al proyecto desarrollado en la que se explica que hace el sistema a nivel general. Posteriormente se ha analizado si el sistema requiere de usuarios y si se lleva a cabo un proceso de registro de estos. En caso de que se lleve a cabo el registro de datos personales del usuario, aunque sea simplemente un nombre de usuario y contraseña, se comprueba si a lo largo del documento existe alguna sección o capítulo que especifique los datos que se toman y si existen requisitos de seguridad que cubran la confidencialidad de estos. Adicionalmente, ya que

se está tratando con datos personales en muchos de los casos se ha tenido en cuenta la aparición de un apartado que haga referencia a la legislación aplicable al sistema, como puede ser la LOPD. En este caso no se ha tenido en cuenta la actual RGPD ya que ésta entró en vigor a finales del año 2017 y no hay TFG analizados posteriores a esa fecha. Finalmente, se argumenta si el autor del proyecto ha tenido en cuenta los mecanismos de seguridad necesarios para proteger las credenciales de usuario o si no se han tenido en cuenta. Para poder asignar un valor cuantitativo a este grado de seguridad provisto por el autor, se ha decidido hacer una escala del cero al cuatro. Los niveles de la escala de seguridad se explican a continuación:

- **Grado 0:** este grado es el más bajo de todos y significa que a pesar de que el proyecto trata con datos de usuarios y requiere del uso de las credenciales de usuario, el autor omite completamente la seguridad y no hace requisitos de seguridad ni referencia a las legislaciones aplicables al sistema.
- **Grado 1:** en este grado de seguridad, el autor hace al menos mención o a unos requisitos de seguridad debido al tratamiento de datos de usuarios o se menciona de forma somera la legislación aplicable. Con que se dé una de las dos opciones se obtiene este grado de seguridad.
- **Grado 2:** este grado de seguridad es igual que el Grado 1, con la diferencia que es necesario que se mencionen ambas opciones en el documento del proyecto.
- **Grado 3:** este grado de seguridad amplía el Grado 2 de seguridad, de forma que se espera que se especifique la forma de la que se va a implementar la seguridad mencionada en los requisitos de seguridad o si se ha realizado un análisis en detalle del marco regulador o legal.
- **Grado 4:** este es el último grado y es el que se debería aspirar a tener, en este grado se hace mención de los requisitos de seguridad y se argumenta las decisiones de los mecanismos de seguridad empleados o se especifica qué algoritmo se empleará. Adicionalmente, se hace un análisis en profundidad del marco regulador o marco legal.

El grado de seguridad de cada uno de los TFG se especifica al final del análisis de cada uno de ellos.

TFG 1: "Herramienta de análisis de legibilidad de contenidos educativos" [18]

El objetivo de este TFG es el análisis de la legibilidad de contenidos educativos a través del método *Gunning Fog*. Para ello, se desarrolla un sistema que poseerá usuarios para que puedan subir los textos y el sistema los analice.

Primero hay que mencionar que los datos que se requiere almacenar en el sistema sobre los usuarios son los siguientes:

- Nombre: que es el nombre real del usuario.
- Apellidos: que son los apellidos reales del usuario.
- Email: empleado como identificador de usuarios dentro del sistema.
- Contraseña: para poder iniciar sesión en el sistema.
- ID_usuario: id para uso del sistema, este campo no se le pide al usuario.

Estos datos pertenecen a una persona real y por tanto deberían estar protegidos. Respecto a esto, el autor crea dos requisitos de seguridad uno en el que especifica que sólo cuando un usuario inicie sesión se cargarán los datos relativos al mismo y otro en el que especifica que las contraseñas deberán ser cifradas.

El primer problema recae en que el autor no especifica para qué se van a recabar esos datos personales de cada usuario, cosa que debería especificar en su formulario de registro.

Además, los datos de entrada en los casos de uso están poco detallados. En el caso de uso para el *login* de usuarios, si el usuario introduce unas "credenciales incorrectas" se muestra un mensaje en el formulario especificando que los datos son incorrectos. En este caso se debería especificar si en el caso de uso se ha introducido mal el nombre de usuario o contraseña, ya que una estrategia para conocer si existe un usuario con cierto nombre de usuario registrado en el sistema, y por ende tiene una cuenta, es intentar hacer *login* con un nombre de usuario y esperar un mensaje de usuario incorrecto. Tampoco se especifica si el mensaje dice si es la contraseña o el usuario lo que no ha sido introducido correctamente, por tanto, se debería haber sido más específico a la hora de definir los datos de entrada y a la hora de especificar cuál es el mensaje de error, que debería ser un mensaje genérico del tipo "Datos introducidos incorrectos" para evitar darle pistas a un posible atacante.

El siguiente problema recae sobre la forma en la que se almacenan las credenciales de usuario. El segundo requisito de seguridad especifica que las contraseñas serán cifradas, pero no especifica cómo van a ser cifradas. No se menciona si es cifrado simétrico o asimétrico ni el nombre del algoritmo, como puede ser AES-256 para cifrado simétrico. Por tanto, tampoco existe un razonamiento del porqué se escogería uno u otro método de cifrado. En adición a todo esto para el almacenamiento seguro de las contraseñas se deberían emplear funciones hash criptográficamente seguras que permitan el uso de un *salt* criptográfico fuerte. Esta opción tampoco ha sido barajada por el autor.

En caso de que el sistema se viera atacado y se filtrara información sobre los usuarios otro problema es que sólo se menciona el cifrado de las contraseñas y nada respecto a los datos personales de los usuarios que por tanto estarían en claro y en manos del atacante.

Adicionalmente, no se hace referencia en ningún apartado a la legislación aplicable al sistema. El único momento que se hace referencia a la LOPD es en la sección de acrónimos y en ninguna otra sección del documento se menciona.

En definitiva, el autor no ha sido capaz de proveer una serie de mecanismos de seguridad lo suficientemente fuertes para proteger las credenciales y datos de sus usuarios.

El grado de seguridad que se le ha asignado a este TFG es el grado 1, debido a que por lo menos se han mencionado el cifrado de las contraseñas de los usuarios.

TFG 2: "Modelado de un puerto marítimo con Unity 3D" [19]

El objetivo de este TFG es el modelado de un puerto marítimo empleando la herramienta Unity 3D para que un usuario en la simulación pueda ver el escenario con distintas condiciones atmosféricas. Dichas condiciones atmosféricas pueden ser modificadas por el usuario para comprobar las distintas situaciones con diversas embarcaciones con unas características atmosféricas específicas.

El autor deja claro que el simulador no posee ningún tipo de guardado de los datos de cada una de las sesiones y que todos los datos generados en cada simulación se deben borrar y no ser almacenados. Adicionalmente, hace referencia a la LOPD sobre la ausencia de usuarios y datos sobre los mismos por lo que el sistema no debe tener ningún tipo de protección de credenciales de usuario ni de datos personales.

El autor ha sido capaz de mantener el sistema seguro, en lo referente a las credenciales, debido a la ausencia del almacenamiento de datos sobre usuarios y su política de borrado de información en la finalización de cada una de las sesiones de simulación.

El grado de seguridad que se le ha asignado a este TFG es el grado 2 debido a que ha sido capaz de explicar que no se almacena información alguna de las sesiones, ni de los usuarios que hagan uso de la aplicación y la referenciación al efecto de la LOPD sobre el sistema desarrollado.

TFG 3: "Desarrollo de una aplicación multimodal para apoyar el estudio mediante m-learning" [20]

Este TFG se centra en la creación de una aplicación desarrollada en Android para la modalidad de m-learning. Se centrará en apoyar al usuario, que serán alumnos matriculados en asignaturas, a asentar los conocimientos de las asignaturas matriculadas mediante pruebas de diez preguntas tipo test. Las preguntas de tipo test pueden responderse de forma oral o mediante la pantalla táctil. Esta aplicación requiere del registro y almacenamiento de usuarios para que posteriormente puedan identificarse con el sistema para poder realizar las pruebas de conocimiento de las asignaturas.

Respecto el almacenamiento de las credenciales de usuario solamente se menciona que se guardan en la base de datos MySQL una vez se realiza el registro. El sistema emplea un servicio llamado Hostinger que le ofrece un servicio de 2 bases de datos MySQL.

El documento no presenta de por sí ningún requisito de seguridad, presumiblemente debido a que emplea el servicio de Hostinger y ha decidido emplear la seguridad de punto a punto que proveen mediante certificados SSL. En caso de que la máquina física o la compañía de Hostinger se viera afectada por un ataque, el autor no ha provisto de ningún método de protección de credenciales. Para ello, debería haberse especificado el uso de funciones hash criptográficamente seguras con el uso de *salts* generados a través de un CSPRNG con una longitud suficiente para el hasheado de la contraseña y su posterior almacenamiento.

Otro problema encontrado es que en el esquema de *login* de un usuario en la aplicación, las credenciales de usuario se almacenan en las *SharedPreferences*. Con lo cual, las credenciales se encuentran almacenadas en el dispositivo en claro. Esta decisión no parece la más segura ya que podrían ser potencialmente atacadas y robadas. Una mejor opción para el protocolo que emplea el autor para el *login* de un usuario sería recibir un código de respuesta del servidor web para saber si las credenciales coinciden con las que están almacenadas en la base de datos y así no transferir de forma innecesaria y poner en riesgo las credenciales de usuario.

En el apartado del marco regulador el autor hace referencia a la legislación que afecta al sistema, entre otras la LOPD.

El autor propone en los trabajos futuros aumentar la seguridad de la aplicación empleando técnicas de ofuscación de código y cifrado en las bases de datos MySQL. Aun así, a la hora de diseñar el sistema no se ha tenido en cuenta la seguridad de las credenciales de usuario.

El grado de seguridad que se le ha asignado a este TFG es el grado 1 porque se ha realizado un estudio justificando las leyes que pueden afectar al sistema.

TFG 4: "Aplicación nutricional y seguimiento deportivo, en atletas de alto rendimiento" [21]

El proyecto es una aplicación web nutricional y seguimiento deportivo para atletas en la que el sistema distingue entre administrador, entrenadores y usuarios. Se ofertan distintos servicios como rutinas, ejercicios y una nutrición para los usuarios.

En el proyecto existe un apartado con unos requisitos de seguridad. En ellos se menciona el cifrado de las contraseñas y comprobaciones con JavaScript que se entiende serían para evitar ataques de inyección de SQL y *Cross Site Scripting*. Además, existen otros requisitos que especifican que los datos de los clientes emplearán un sistema de cifrado y que el sistema se desarrollará para evitar cualquier tipo de ataque informático. Más allá de las especificaciones en los requisitos no se provee más información respecto a los sistemas de cifrado que se supone que empleará el sistema para cifrar los datos de los clientes. Tampoco se explica cómo se cifran las contraseñas de los usuarios ni el cómo se llevarán a cabo las comprobaciones en JavaScript para defenderse frente a posibles ataques. Adicionalmente, no se provee información de cómo se diseña el sistema para evitar cualquier tipo de ataque informático.

El autor no añade una justificación para la decisión del cifrado de las contraseñas ni un estudio de las alternativas existentes como es el uso de hashes y *salts* criptográficamente fuertes.

Respecto a los datos de los usuarios, en el manual de usuario se puede apreciar que se piden información personal la cuál no se especifica cómo va a ser protegida.

Adicionalmente, no se hace referencia en ningún apartado a la legislación que puede afectar al sistema.

En definitiva, el autor intenta tener en cuenta la seguridad a la hora de diseñar el sistema, pero simplemente menciona que se tomarán acciones respecto a ella y no especifica ninguna solución ni los algoritmos de cifrado que se van a emplear.

El grado de seguridad que se le ha asignado a este TFG es el grado 1 debido a que se hace mención al cifrado de las contraseñas.

TFG 5: "Aplicación móvil para la detección de pasos basada en acelerometría a baja velocidad" [22]

El TFG se centra en el desarrollo de un algoritmo para distinguir los pases del usuario cuando camina a baja velocidad y poder almacenar los pasos realizados del usuario para poder ofrecer un histórico de la actividad del usuario.

La aplicación en si no ofrece ningún perfil de usuario, es decir, que no requiere del inicio de sesión ni registro previo. En cuanto a los datos almacenados son los pasos realizados ordenados por fecha que pueden ser consultados en el histórico y realmente no poseen información crítica del usuario.

Adicionalmente el autor menciona en el marco regulador la LOPD sobre la recogida de datos de los usuarios y que la aplicación se limita a recoger los datos de su actividad de movimiento o pasos. Por tanto, el autor al no recoger datos personales de los usuarios ni obligar ningún tipo de registro para el uso de la aplicación, ha conseguido que la aplicación sea segura para los usuarios.

El grado de seguridad que se le ha asignado a este TFG es el grado 2 ya que se hace un estudio de cómo afecta la legislación de la LOPD al sistema. Además, no es necesario proteger datos de los usuarios ya que estos no se almacenan.

TFG 6: "Diseño e implementación de un sistema de control remoto para vigilancia en Android" [23]

El TFG se centra en el desarrollo de un sistema de videovigilancia para la plataforma Android. La aplicación desarrollada en Android Studio requerirá de un registro previo de los usuarios del sistema.

En el registro de los usuarios se piden los siguientes campos:

- Nombre de usuario
- Contraseña
- Código de usuario que es un código proporcionado por el sistema para identificar el sistema de videovigilancia
- Correo electrónico
- Dirección Postal

En el documento no se hace referencia a ningún requisito de seguridad para el sistema a desarrollar. La única mención a la seguridad es en el apartado del diseño físico del sistema en el que el autor explica que la contraseña, el correo electrónico y la dirección postal del usuario se cifrará con SHA-256. En esta afirmación yace el primer problema, debido a que SHA-256 es una función hash perteneciente a la familia SHA-2 desarrollada por la NSA y no un algoritmo de cifrado y descifrado. El uso de esta función hash sería correcto para el hasheado de las contraseñas siempre y cuando se le aplique un *salt* criptográficamente fuerte para evitar ataques de diccionario, *lookup tables* y *rainbow tables* en caso de que la base de datos se viera comprometida.

Respecto al hasheado del correo electrónico y el código postal no sería una solución correcta, ya que son datos que no se quieren hashear por que no se puede invertir el resultado de un hash y presumiblemente el correo electrónico y código postal se almacena para un posible contacto con el usuario. La solución sería aplicar un sistema de cifrado a esos campos. El tipo de cifrado

se debería elegir dependiendo de la frecuencia con la que se espera contactar con los usuarios correspondientes. En caso de que sea algo bastante común el enviar correos electrónicos a los usuarios un cifrado simétrico sería el más adecuado debido a que es más rápido que el cifrado asimétrico. En caso de que estas comunicaciones con el usuario fueran menos frecuentes el uso de un sistema de cifrado asimétrico sería el más recomendable debido a que se almacena el código postal del usuario y su correo electrónico, datos mediante los cuales se puede llegar a saber dónde vive y conocer su identidad.

Teniendo en cuenta esto último, el autor tampoco hace referencia a la LOPD a pesar de estar pidiendo datos personales y críticos a los usuarios. Tampoco se especifica el motivo por el cual se pide al usuario el código postal.

Se puede concluir que el desarrollo del sistema no ha tenido en cuenta la seguridad de las credenciales de usuario. Ha habido un intento por cifrar las contraseñas y datos son personales de los usuarios, pero el autor ha confundido conceptos de cifrado y hasehado. Aunque idealmente el autor se refiriera al hasheado de las contraseñas con SHA-256, no se emplean *salts* generados por los CSPRNG y las credenciales siguen siendo susceptibles a ataques.

El grado de seguridad que se le ha asignado a este TFG es el grado 1 debido a que ha intentado proteger las contraseñas de los usuarios mediante SHA-256.

TFG 7: "Desarrollo de gestor de comidas a domicilio mediante la aplicación de mensajería instantánea Telegram" [24]

El objetivo del proyecto es desarrollar un Script para Telegram implantado en un sistema formado por una base de datos y un SO alojado en una Raspberry Pi. El sistema se espera que se instale en distintos establecimientos para la realización de pedidos a domicilio.

Para el funcionamiento del sistema el usuario debe comunicarse con el sistema a través de Telegram. El sistema tiene la capacidad de distinguir si el número con el que está comunicándose es uno de los almacenados en la base de datos, si se da este caso se trata de un trabajador del establecimiento. O si se trata de un número no registrado en cuyo caso el sistema asume que se trata de un cliente normal del establecimiento. El sistema como tal no posee la necesidad de la creación de cuentas de usuario para identificarlos, por tanto, no se almacenan credenciales de usuarios en el sistema. Aun así, entre los datos que sí que se almacenan en la base de datos sobre los pedidos realizados se encuentran el número de teléfono de los usuarios y la dirección a donde se realiza el envío, que presumiblemente es dónde vive el cliente.

El documento posee un requisito de seguridad que especifica que los datos de los consumidores estarán cifrados en la base de datos. A parte de este requisito no existe ningún requisito adicional o apartado en el documento que en el que se especifiquen los datos exactos a cifrar ni el tipo de cifrado que se empleará.

Adicionalmente, en el apartado del marco regulador se hace mención de la LOPD y a la directiva 2002/58/CE de Privacidad y Comunicaciones Electrónicas (*European Comision 2013*). A continuación, se aclara que el desarrollo del sistema protegerá los datos de los usuarios cifrando el medio de almacenamiento en el que se encuentren, usar los datos para el fin con el que se anunció a los usuarios y proveer los derechos ARCO (Acceso, Rectificación, Oposición y Cancelación) a los usuarios. Otra vez, se vuelve a mencionar el cifrado del medio donde están almacenados los datos, pero no se menciona la forma en la que se llevará a cabo. No se ponderan las posibilidades de usar un cifrado simétrico o asimétrico ni las razones para usarlo.

Se puede asegurar que el desarrollo de la aplicación intentó tener en cuenta la seguridad de los datos de los clientes de los restaurantes en los que se instale el sistema. Aun así, el proyecto falla a la hora de especificar qué tipo de cifrado y cómo se llevará a cabo. Adicionalmente, se delega la responsabilidad de la gestión del sistema al restaurante en el que se instale el dispositivo, así como el deber de informar a los clientes sobre la información que se recabará sobre ellos.

El grado de seguridad que se le ha asignado a este TFG es el grado 2 ya que el autor menciona el cifrado de los datos personales y se hace un estudio de la legislación que afecta al sistema.

TFG 8: "Desarrollo de un asistente multimodal educativo para dispositivos móviles Android" [25]

El objetivo del TFG es la creación de una aplicación para dispositivos móviles Android en la que se podrá desarrollar la metodología educativa del *mobile learning*. Para el uso de la aplicación el usuario deberá registrarse y posteriormente iniciar sesión para identificarse en el sistema. No existe un apartado con los requisitos del sistema, en su lugar solo existen casos de prueba. A la hora de registrarse un usuario debe introducir los siguientes campos:

- Nombre de usuario
- Dirección de e-mail
- Confirmación de e-mail
- Contraseña

Estos datos se persisten realizando una conexión Android Studio – MySQL desde la capa de lógica de negocio a la capa de persistencia. La base de datos escogida por el autor es MySQL y su uso está justificado en uno de los apartados.

En ningún momento se hace referencia a ningún concepto de seguridad. No existe explicación si se almacenan los datos en claro o cifrados en la base de datos. Tampoco se expone el posible uso de funciones hash para el almacenamiento de las contraseñas ni el uso de *salts* criptográficamente fuertes. Adicionalmente, no se hace referencia al marco regulador ni a las leyes correspondientes como puede ser la LOPD.

El grado de seguridad que se le ha asignado a este TFG es el grado 0 ya que el desarrollo del sistema no ha tenido en cuenta en absoluto la seguridad de las credenciales de usuario ni las leyes que lo regulan.

TFG 9: "Diseño e implementación de un sistema domótico basado en Raspberry Pi" [26]

El desarrollo del sistema se centra en la creación de un sistema domótico empleando una Raspberry Pi y un servidor web Apache. El sistema permitirá al usuario consultar temperatura y humedad de habitaciones, controlar la apertura de puertas y persianas, consultar histórico de temperaturas, consultar movimientos en una habitación y controlar la iluminación interior. Una de las restricciones del sistema es que no se podrá acceder desde internet y por tanto el usuario debe estar conectado a la misma subred para poder emplear el sistema.

En cuando al sistema no se requiere el registro de los usuarios, sino que viene con un usuario predeterminado. Este usuario mostrado en una captura emplea un email y contraseña para el acceso, presumiblemente dichos datos están guardados en la base de datos MySQL para el proceso de autenticación. En la captura que se muestra el email es admin@admin.es y la contraseña consta de ocho caracteres. De forma altamente probable la contraseña sea *password*

que coincide con el límite de ocho caracteres o fácilmente sea una de las contraseñas más empleadas que puede ser fácilmente comprobada en una *password dumpsite*.

En este aspecto el autor no ha tenido demasiado en cuenta la seguridad del sistema debido a que ha empleado unas credenciales demasiado evidentes para el control del sistema. Si bien es cierto que sería necesario que el atacante estuviera conectado a la misma subred. Adicionalmente, los datos almacenados en la base de datos, que son principalmente históricos de medidas del sistema, no se menciona si están cifrados o no. Al no ser datos excesivamente críticos y al tener la limitación de no poder acceder desde internet este aspecto no es tan grave. Por último, mencionar que no se hace mención de ningún tipo de marco regulador sobre la información recogida con los sensores en la vivienda del cliente.

El grado de seguridad que se le ha asignado a este TFG es el grado 0 debido a que, a la hora de desarrollar el proyecto no se ha tenido en cuenta la seguridad del sistema y credenciales de usuario, presumiblemente debido a la limitación de no poder acceder desde internet.

TFG 10: "Simulación de trayectorias de barcos y aplicación al control marítimo" [27]

Este proyecto tiene el objetivo de desarrollar un software para la simulación de trayectorias de barcos para aumentar la seguridad en el mar y ayudar a los controladores navales.

El sistema no posee usuarios y por tanto no existe un proceso de autenticación ni de almacenamiento de credenciales de usuario. Existen requisitos de seguridad que especifican que la información sobre cada simulación debe eliminarse al finalizar la ejecución y no almacenarse en ningún caso.

En este caso el autor ha optado por la creación del software sin necesidad de la creación de un sistema que emplee usuarios. El sistema en sí no recoge ningún dato del usuario que emplea la herramienta y adicionalmente los datos de la sesión se eliminan al terminar la ejecución, por tanto, el autor ha conseguido tener en cuenta la seguridad del sistema. No tanto por proveer sistemas de seguridad para la protección de datos, pero si debido a la omisión o falta de necesidad de datos de los usuarios.

El grado de seguridad que se le ha asignado a este TFG es el grado 1 debido a que se ha especificado el borrado de los datos de las simulaciones y por ende ningún dato es almacenado.

TFG 11: "Estrategias de diversificación eficiente de carteras e implementación de una plataforma digital de inversión" [28]

El proyecto se centra en desarrollar una plataforma digital de inversión para los usuarios. Se centra en la diversificación de carteras y permitir al usuario realizar predicciones estocásticas sobre activos. Para ello el sistema requiere de un registro previo en el que se toman los siguientes datos:

- Nickname
- Email
- Name
- Lastname
- Country
- Password
- Confirm Password

Todos estos datos son los que se exigen además de una casilla en la que se acepta la recolección de dichos datos en la que se especifica que el proceso se realiza de acuerdo con la LOPD.

Respecto al aspecto de seguridad, el documento cuenta con ciertos requisitos de seguridad, pero no se comenta nada relacionado con el tratamiento de los datos del usuario y menos con sus credenciales. Más adelante en el documento en un apartado se exponen distintas tecnologías que emplean varias empresas para cubrir los requisitos de confidencialidad de los datos, pero aun así no se concreta si se va a emplear alguno de ellos. Otra aparición respecto a la protección de las credenciales de usuario es en la explicación de las Bean para el acceso a la base de datos en la que se especifica que la contraseña de usuario se encuentra codificada mediante MD5. En la descripción del LoginServlet del sistema también aparece en la función de validar la contraseña el procedimiento de autenticación del MD5 de la contraseña recibida por parámetro y la comparación de ese resultado con el valor almacenado en la base de datos.

En el documento también existe un apartado específico para el marco regulador aplicable sobre el sistema a desarrollar y una justificación argumentada de que leyes son aplicables al proyecto, entre ellas la LOPD debido a la recogida de datos personales de los usuarios.

En definitiva, se puede apreciar que el autor ha tenido en cuenta la seguridad a la hora de desarrollar el sistema. Se han realizados requisitos de seguridad, se ha empleado una función hash, aunque la función MD5 sea susceptible a ataques de colisiones y no se mencione el uso de *salts*. También se ha contemplado las leyes que afectan al desarrollo del sistema y se han tenido en cuenta.

El grado de seguridad que se le ha asignado a este TFG es el grado 2 debido a que se ha empleado la función hash MD5 y la mención de la legislación aplicable.

TFG 12: "Desarrollo de una aplicación de turismo gastronómico para dispositivos iOS y análisis estadístico" [29]

El TFG tiene como objetivo la creación de una aplicación de turismo gastronómico para iOS. En dicha aplicación los usuarios pueden buscar platos típicos de la zona, buscar que restaurantes los ofrecen y hacer valoraciones sobre los mismos.

El sistema requiere de un registro de usuarios y de autenticación para el uso de la aplicación. Existe un único requisito de seguridad en el que se especifica que el manejo y almacenamiento de las contraseñas debe ser siempre de manera cifrada y que esto lo llevará a cabo Parse.

El autor delega la responsabilidad de la seguridad de las credenciales de usuario en Parse sin mencionar que ventajas o métodos de cifrado se ofrecen. Tampoco se hace una argumentación por la cual se explique el porqué del cifrado de las contraseñas y no el uso de una función hash con un *salt* correspondiente. Adicionalmente, el autor hace referencia a la LOPD en el apartado del marco regulatorio por lo que se entiende que asegurará la confidencialidad de los datos de los usuarios.

El grado de seguridad que se le ha asignado a este TFG es el grado 2 debido a que el autor ha sido capaz de delegar la responsabilidad del mecanismo de seguridad para la protección del almacenamiento de los datos de los usuarios a Parse, así como tener en cuenta la LOPD a la hora de desarrollar el proyecto.

TFG 13: "Diseño e implementación de Campus Life, una aplicación móvil para la consulta e inscripción de actividades deportivas y culturales" [30]

El TFG tiene como objetivo el desarrollo de Campus Life que es una aplicación compatible con iOS para solventar las necesidades del Espacio de Estudiantes de la UC3M. La aplicación permite hacer consultas e inscripciones en actividades deportivas, culturales y de orientación. El acceso a la aplicación está limitado a los estudiantes, profesores y personal de administración y servicios de la UC3M.

El sistema posee usuarios con sus respectivas credenciales. El documento no posee requisitos de seguridad que especifiquen como se deben almacenar los datos de los usuarios. En el apartado de la definición de la base de datos se especifica la entidad usuario en la que el campo pass, que es la contraseña, se especifica que está cifrada pero no cómo. Adicionalmente, no se hace referencia al marco regulador ni a las leyes que pueden afectar al sistema como puede ser la LOPD.

El grado de seguridad que se le ha asignado a este TFG es el grado 0 ya que el autor no ha sido capaz de tener en cuenta la seguridad al desarrollar el sistema ni las leyes que pueden afectar al sistema.

TFG 14: "Desarrollo de una aplicación móvil en el ámbito deportivo" [31]

El TFG desarrolla una aplicación móvil deportiva para crear un nuevo entorno para personas que tienen dificultades con aplicaciones tradicionales de deportes. El sistema puede recibir entradas táctiles o por voz facilitando su uso por parte de gente con discapacidades.

El sistema requiere de usuarios que deben registrarse y posteriormente autenticarse con el sistema para acceder a los servicios que ofrece. Los datos requeridos en el registro son:

- Nombre
- Email
- Ciudad
- Password

A lo largo del documento no existen requisitos de seguridad que especifiquen el tipo de cifrado o mecanismo de seguridad que se va a emplear para asegurar la confidencialidad de los datos de los usuarios. Tampoco existe ninguna otra alusión en el documento a cualquier tipo de cifrado o mecanismo de seguridad. Tampoco se hace referencia en ningún apartado a las leyes que pueden afectar al sistema como es la LOPD.

El autor no ha tenido en cuenta la seguridad en el desarrollo de la aplicación a pesar de necesitar usuarios y almacenar información sobre sus credenciales. Tampoco ha tenido en cuenta las leyes del marco regulador en el momento que se desarrollaba la aplicación.

El grado de seguridad que se le ha asignado a este TFG es el grado 0.

TFG 15: "Diseño y desarrollo de un portal eCommerce de ropa, calzado y accesorios deportivos" [32]

El proyecto desarrolla un portal de *eCommerce* para poder vender y comprar ropa calzado y accesorios deportivos. El portal posee distintos tipos de usuarios, entre ellos el administrador, los proveedores y los usuarios (compradores). El sistema ofrece un sistema de registro y de inicio

de sesión, por ello los datos del registro son almacenados en una base de datos. Los campos que se le piden al usuario al registrarse son:

- Nombre
- Apellidos
- Descripción
- Dirección
- Código postal
- Municipio
- Ciudad
- País
- Email
- Contraseña
- Telefono

Como se puede apreciar son varios los datos que se piden al usuario y muchos de estos son datos personales. Estos datos son posteriormente almacenados en la base de datos en la tabla Users. A lo largo del documento no existen requisitos de seguridad que especifiquen el mecanismo de seguridad para proteger los datos del usuario. Sin embargo, en el apartado en el que se define la tabla Users se especifica que el campo *Password* almacenará la contraseña del usuario haciendo uso de una función de cifrado. En cuanto a las leyes que puedan afectar al proyecto no se menciona ninguna ni existe un apartado del estilo marco regulador.

Se puede concluir entonces que el autor no ha tenido en cuenta la seguridad a la hora de desarrollar el sistema. Se ha mencionado el uso de una función de cifrado, pero no se ha especificado y no se ha justificado sobre el uso de una función hash con *salt*. Tampoco se ha tenido en cuenta las posibles leyes que afectan al sistema debido a la cantidad de datos personales de los usuarios.

El grado de seguridad que se le ha asignado a este TFG es el grado 1 puesto que se menciona el uso de la función de cifrado.

TFG 16: "Exam developer: desarrollo de una aplicación móvil enfocada a m-learning" [33]

El desarrollo del proyecto se centra en el desarrollo de una aplicación para Android en la que se podrá realizar exámenes. Dichos exámenes están constituidos de ejercicios que permitirán ayudar a personas con Alzheimer y TDA. Además de poseer ejercicios similares a los propuestos en las entrevistas de trabajo más comunes.

La aplicación requiere de un registro e inicio de sesión previo para poder interactuar con el sistema de exámenes. Los datos que se le piden al usuario son:

- Name
- Age
- Username
- Password

Estos datos se almacenarán en una base de datos MySQL a la que se accederá a través de un servidor web que está desarrollado empleando PHP.

En el documento no existe un apartado de requisitos, por tanto, no existe ningún requisito que tenga en cuenta la confidencialidad de las credenciales de los usuarios. Tampoco existe ninguna referencia a la forma en la que se almacenan los datos de los usuarios. No hay mención ni siquiera del cifrado de los datos en la base de datos. En cuanto a las leyes que puedan afectar al sistema no existe ningún apartado o mención de la LOPD. Lo único a favor del desarrollo de la aplicación es que el mensaje de error de inicio de sesión no especifica si el campo incorrecto es el usuario o la contraseña.

El grado de seguridad que se le ha asignado a este TFG es el grado 0 debido a que el autor no ha tenido en cuenta la seguridad a la hora de desarrollar el proyecto y mucho menos la confidencialidad de las credenciales de usuario.

TFG 17: "Desarrollo de una plataforma social para el suministro colaborativo de piezas de repuesto" [34]

El TFG tiene el objetivo de desarrollar una aplicación Android para reactivar el mercado de las piezas usadas de automóviles de forma que a través de la aplicación la comunicación entre los clientes y proveedores sea sencilla. En la aplicación se podrán publicar anuncios que aportarán información sobre la pieza en cuestión para posibles compradores. Para el uso de esta aplicación es necesario hacer un registro para crear una cuenta y poder autenticarse en el sistema. Los datos que se requieren del usuario son los siguientes:

- Nombre de usuario
- Email
- Contraseña
- Comprobación de contraseña
- Dirección

Estos datos son almacenados en una base de datos MySQL. En el apartado del marco regulador se hace alusión a la LOPD y la necesidad de cumplirla debido a la recogida de estos datos. Adicionalmente, el autor especifica que se almacenarán el hash de las contraseñas y no las contraseñas en claro. La función hash que empleará es la función bcrypt. Además, se hace alusión al Artículo 36 de la Ley 32 general de Telecomunicaciones del 3 de noviembre de 2003 que especifica que la información transmitida por internet puede ser protegida a través de mecanismos de cifrado. En este caso el autor empleará el protocolo HTTPS para el intercambio de datos entre el cliente y el servidor. También, existe un requisito no funcional de seguridad en el que se especifica el uso del protocolo HTTPS para las comunicaciones entre el cliente y el servidor.

El grado de seguridad que se le ha asignado a este TFG es el grado 3 puesto que el autor sí que ha tenido en cuenta la seguridad a la hora de desarrollar el sistema, así como las leyes que pueden afectar a este. El autor además escoge el uso de la función hash *bcrypt* para el hasheado de las contraseñas. Esta función permite el uso de *salts* y especificar un número de iteraciones para reducir el riesgo de ataques de fuerza bruta. El único aspecto que no ha cubierto es si se emplearán *salts* y cómo se generarían los *salts* criptográficos. Otro punto que podría haberse cubierto es el cifrado de los datos del usuario en la base de datos para añadir un poco más de seguridad.

TFG 18: "WeSweat, Geolocalización social de deportistas" [35]

El proyecto se centra en la creación de la empresa WeSweat S.L. cuyo objetivo es crear una gama de productos para simplificar la creación de eventos deportivos a través de la generación de redes de deportistas. Para el uso de la aplicación se necesita cumplimentar un formulario de registro para generar una cuenta y posteriormente poder autenticarse en el sistema.

En el documento a la hora de especificar los requisitos no funcionales se enumera que existen requisitos de seguridad, pero al proceder a buscarlo en las tablas correspondientes no aparece ningún requisito de seguridad. En el documento hay un apartado dedicado al marco legal, especialmente a la protección de datos. En este apartado se hace referencia a la LOPD y por tanto registrar la actividad de la empresa en la AEPD. Se menciona la recogida de datos de terceros como Facebook en caso de que el usuario se registre con la cuenta de Facebook.

Se puede concluir que el autor no ha tenido en cuenta la seguridad de las credenciales de usuario a la hora de desarrollar el sistema. Aunque, en los trabajos futuros se propone añadir el cifrado de las comunicaciones entre servidores y el cifrado de datos de almacenados.

El grado de seguridad que se le ha asignado a este TFG es el grado 1 debido al estudio de la legislación aplicable.

TFG 19: "Parkineo, aplicación Android para la búsqueda de parking" [36]

El objetivo del proyecto es desarrollar una aplicación en Android para encontrar parkings cercanos a la posición del usuario. Además, la aplicación permitirá saber si los parkings están llenos o hay sitio para aparcar.

En la aplicación se pueden registrar usuarios como usuarios normales o usuarios autorizados. Independientemente, para el inicio de sesión se requiere el uso de las credenciales de usuario establecidas en el registro. A lo largo del documento no aparece ningún requisito de seguridad o apartado en el que se especifique como se asegurará la confidencialidad de las credenciales de usuario.

Por otra parte, existe el apartado del marco regulador en el que se hace referencia a la LOPD debido a la recogida de datos de los usuarios a la hora de hacer el registro.

En definitiva, el autor no ha tenido en cuenta la seguridad de las credenciales de usuario del sistema desarrollado. No se han propuesto mecanismos de seguridad para asegurar la confidencialidad de los datos de los usuarios, aunque se haya mencionado la aplicación de la LOPD.

El grado de seguridad que se le ha asignado a este TFG es el grado 1 puesto que el autor ha mencionado la legislación aplicable al sistema.

TFG 20: "Entorno y aplicación móvil Android "HISPANIA UC3M"" [37]

El proyecto tiene el objetivo de desarrollar una aplicación móvil junto con un backend para la adquisición de conocimientos mediante una estructura gamificada. La herramienta desarrollada tiene una naturaleza pedagógico-social y pretende llevar a cabo una integración de cuentas en redes sociales que aporten valor añadido a la gamificación. Para el uso de esta aplicación es necesario un registro previo. El registro puede realizarse a través de un formulario o mediante una cuenta de Facebook.

En el caso de que el usuario escoja la opción del formulario, se le piden los siguientes datos:

- Email
- Contraseña
- Repetición de la contraseña
- País

Los datos que se requieren del usuario son pocos, pero críticos. El autor ha tenido en cuenta la necesidad de la protección de la contraseña y existe un requisito de software que lo especifica. En la descripción del requisito se especifica que la contraseña debe ser cifrada como mínimo a MDM-5. Suponiendo que se trate de una errata, el autor puede estar refiriéndose a la función hash MD5. Aunque ese fuera el caso existiría un error de concepto ya que MD5 es una función hash y no de cifrado. También se especifica en el apartado que describe el modelo de datos en el *backend* que el modelo de datos lógicos se ha realizado sin preocupación de cómo se almacenan los datos internamente en el *backend*. Esta no es razón suficiente para no hacer un análisis y justificación de cómo se deben de tratar los datos sensibles, como el cifrado de datos del correo electrónico. Adicionalmente, a lo largo del documento no se hace referencia a ninguna ley de protección de datos o que afecte al sistema.

El grado de seguridad que se le ha asignado a este TFG es el grado 1 ya que el autor intentó tener en cuenta la seguridad a la hora de diseñar el sistema. Se mencionaron requisitos de seguridad, pero no se ha mencionado nada relativo al marco regulatorio que afecta al proyecto.

TFG 21: "Aplicación de comercio electrónico para pequeñas y medianas empresas a través de las tecnologías Open Source" [38]

El objetivo del proyecto es el poder desarrollar un sistema que permita la venta online a nivel mundial, que sea desarrollado a través de tecnologías *Open Source*. Las cualidades que debe aportar el sistema sería la gestión de todas las tiendas, las ventas, facturación, estadísticas y gestión de clientes. Uno de los beneficios del desarrollo de este sistema que se reducirá la necesidad de compra de distintos sistemas para la gestión de los elementos mencionados previamente.

Este sistema permite que los usuarios se registren en el sistema y que inicien sesión. En el documento no se mencionan que datos se le piden al usuario, pero a partir del diagrama de clases del sistema se ha deducido que al menos se requiere:

- Nombre
- Apellidos
- Dirección
- Teléfono

El propio documento menciona que los usuarios registrados podrán iniciar sesión en el sistema mediante de las credenciales de usuario. Por tanto, aunque no se mencione en el diagrama de clases, los usuarios deben tener una contraseña. De hecho, en el apartado de la validación de los requisitos, se menciona el caso del inicio de sesión en el que uno de los pasos para llevar a cabo la prueba es "Se introduce el nombre de usuario y contraseña y se envía".

En cuanto a requisitos software que se refieran a la seguridad del sistema se pueden mencionar que el sistema debe está fortificado ante ataques de *SQL Injection* y *XSS*. Además, las comunicaciones del sistema están protegidas gracias al uso de una conexión segura con SSL/TLS.

En el documento también hay un apartado específicamente dedicado a la LOPD en el que se explica cómo afecta la ley al sistema y se menciona alguno de los mecanismos que se deben seguir para asegurar la confidencialidad de los datos de los usuarios. Uno de los mecanismos mencionados es que el almacenamiento de la base de datos está cifrado.

El autor ha mencionado métodos y las leyes aplicables al sistema desarrollado, pero realmente no ha especificado como llevarlos a cabo. En el caso del requisito que especificaba que se fortificará el sistema contra ataques *SQL Injection* y *XSS*, no se menciona de qué forma se llevará a cabo. Cuando se menciona el cifrado de la base de datos, no se justifica el porqué del cifrado de toda la base de datos. Tampoco se menciona que tipo de cifrado se va a emplear ni el porqué. Adicionalmente, no se hace referencia a la posibilidad de almacenar el valor hash de las contraseñas en vez del cifrado de estas, ni una comparación para justificar por qué el cifrado es mejor opción que el uso de una función hash.

En definitiva, el autor ha tenido en cuenta la seguridad a la hora de desarrollar el sistema, pero no ha llegado a especificar de qué forma se implementará. Adicionalmente, es totalmente consciente de la legislación aplicable al sistema y menciona que se debe hacer para cumplir con ella.

El grado de seguridad que se le ha asignado a este TFG es el grado 2 puesto que el autor hace un estudio de la legislación aplicable al sistema y el cifrado de la base de datos.

TFG 22: "Desarrollo de una aplicación para la gestión de proyectos no gubernamentales" [39]

El objetivo del TFG es desarrollar una aplicación web para la Fundación Madrina. El desarrollo del proyecto se lleva a cabo empleando la metodología SCRUM para poder actualizar el sitio web actual de la fundación.

En la aplicación web existe la posibilidad de registrarse como voluntario a través de la cumplimentación de un formulario. La aplicación web cuenta con un sistema de inicio de sesión en la que se introducen las credenciales de usuario especificadas en el registro. Debido a esta funcionalidad los datos recogidos y almacenados deben ser correctamente protegidos, ya que, pueden ser datos de carácter personal.

En la especificación de los requisitos hay un subconjunto de requisitos que se refieren a la seguridad del sistema y protección de las contraseñas. Se menciona que las contraseñas en la base de datos se almacenan de forma segura y que los datos almacenados cumplirán con la normativa LOPD. Adicionalmente en el apartado del impacto legal se hace referencia a la necesidad de cumplir con las leyes LSSI y LOPD de forma que se asegure la seguridad del sistema para evitar vulnerabilidades. También se hace referencia a la necesidad de que la fundación registre la web desarrollada en la AEPD.

En definitiva, se puede apreciar que la autora ha desarrollado el sistema teniendo en cuenta la seguridad de los usuarios y haciendo un estudio de que legislaciones pueden afectar al sistema. Aun así, muchas de las propuestas de seguridad son vagas y poco específicas. No se especifica qué tipo de cifrado se va a emplear y el porqué de ese cifrado. Tampoco se menciona como se almacenarán de forma segura las contraseñas. No se justifica el porqué del cifrado de las contraseñas en vez de usar funciones hash criptográficamente fuertes y el uso de *salts* criptográficos para evitar posibles ataques sobre los valores hash.

El grado de seguridad que se le ha asignado a este TFG es el grado 2 puesto que el autor hace un estudio de la legislación aplicable al sistema y el cifrado de los datos.

TFG 23: "Elaboración de una aplicación social web 2.0 de notificación de mensajes entre alumnos" [40]

El objetivo del proyecto es desarrollar una aplicación web social para mensajería entre alumnos. Permite que los mensajes puedan ser entre dos alumnos y entre un grupo. Los grupos son proporcionados por el sistema automáticamente, el sistema crea un grupo por cada asignatura con el objetivo de que la aplicación pueda usarse como un recurso docente.

Para el uso de la aplicación es necesario un inicio de sesión previo con las credenciales de usuario. Los usuarios, serán los alumnos de la universidad, por lo que se deduce que las credenciales de usuario serán las mismas que emplean para el sistema de la universidad. De hecho, se menciona que debido a problemas de seguridad no se emplearía el directorio real de la universidad donde están almacenados los datos de los usuarios, sino que se empleará uno local para describir cómo funcionaría la aplicación.

En el documento se mencionan dos requisitos de seguridad, el primero que explica la creación de una sesión de usuario y el tiempo máximo de vida de esta. El otro requisito especifica que las contraseñas estarán cifradas mediante el formato SHA. En cuanto a la legislación, no existe un apartado que estudie las posibles leyes que afecten al sistema.

Se puede comprobar que el autor ha intentado tener en cuenta la seguridad de las credenciales de usuario en el sistema. La especificación del requisito de seguridad que afecta a las contraseñas menciona que se cifrarán, sin embargo, es confuso debido a que el cifrado que se especifica es SHA. El problema es que la familia SHA no es un algoritmo de cifrado, sino que es una familia de funciones hash por sus siglas *Secure Hash Algorithm*. La diferencia entre un hash y el cifrado es principalmente que la función hash en de una única dirección y la operación es irrevertible, pero el cifrado de datos suele tener asociado el descifrado de los mismos, es decir, una función inversa que deshace el cifrado para poder acceder a los datos. Aunque el autor se refiriera al uso de la función hash SHA y almacenar los valores hash para realizar las autenticaciones, no se especifica qué algoritmo de la familia SHA se emplea. Esto último es importante porque varias funciones de esta familia de funciones hash no son criptográficamente seguras y se ha probado que son susceptibles a ataques.

El grado de seguridad que se le ha asignado a este TFG es el grado 1 puesto que el autor hace referencia al cifrado de las contraseñas.

TFG 24: "Desarrollo de una plataforma backend y frontend para la gestión de contenidos" [41]

El objetivo del proyecto es el desarrollo de una aplicación en la que los usuarios, que en este caso son emprendedores, podrán recibir una orientación respecto a su idea de negocio en base a un análisis de las características base de la idea. Además, el sistema ofrecerá la capacidad de analizar a la competencia relacionada con la idea de negocio propuesta e informará de los errores más comunes que se llevan a cabo en el sector en el que se quiere desarrollar la idea.

Para poder acceder al sistema es necesario un registro previo. El inicio de sesión en la aplicación se llevará a cabo mediante un usuario y contraseña previamente especificados en el formulario de registro.

El documento incluye una sección dedicada a los requisitos de seguridad, pero entre dichos requisitos no se especifica nada relacionado con el método de protección de las credenciales del usuario, ni la protección de sus datos en general. El único requisito relacionado con las contraseñas es uno que especifica el mínimo número de caracteres de las contraseñas que debe ser seis.

En el estudio de viabilidad del sistema uno de los apartados se corresponde con el estudio de las restricciones legales que tendría el proyecto en caso de salir al mercado. En este apartado se menciona la LOPD y que se emplearían los datos para el único fin de poder aportar los servicios de la aplicación a los usuarios y en ningún caso para una actividad ilegal.

Se puede concluir que el autor no ha tenido en cuenta la seguridad de las credenciales de usuario a la hora de desarrollar el sistema. No se ha mencionado a lo largo del documento los métodos mediante los cuales se protegerán los datos de los usuarios. El análisis de la legislación aplicable al sistema, en este caso la LOPD, es una simple mención y no se explican los mecanismos que se implementarán para cumplir con la ley.

El grado de seguridad que se le ha asignado a este TFG es el grado 1 debido a que el autor hace un estudio de la legislación aplicable al sistema.

TFG 25: "Sistema backend y frontend para la gestión de nuevas ideas de negocio" [42]

El proyecto se centra en el desarrollo de una aplicación que permitirá a los usuarios evaluar ideas de negocio. La aplicación podrá ser usada por emprendedores para que el sistema les guie y evalúe las ideas de negocio, así como por grandes empresas que no están obteniendo los resultados esperado y no saben el porqué.

Para poder hacer uso de la aplicación es necesario un registro previo en el sistema. Este registro pedirá los siguientes datos a los usuarios:

- Nombre
- Apellido
- Email
- Contraseña
- Repetir Contraseña

Como se puede observar los datos que se le piden al usuario son datos personales que deben ser tratados para asegurar la confidencialidad de estos. En cuanto a las credenciales de usuario, se empleará el email y la contraseña para el inicio de sesión. En el apartado de restricciones generales se menciona que las contraseñas serán cifradas en la base de datos. Los requisitos de seguridad del documento no especifican el método de cifrado que se va a emplear.

En el apartado de especificación de estándares y normas se menciona la aplicación de la LOPD sobre el sistema ya que los datos recogidos son de carácter personal.

En definitiva, se puede comprobar que el autor ha tenido en cuenta que las credenciales de usuario deben estar protegidas. El método mediante el cual el autor las protegerá no es el más indicado y tampoco existe una justificación de la elección del cifrado sobre otras soluciones como el uso de funciones hash. Adicionalmente, el autor es consciente que los datos recogidos son de carácter personal y deben tratarse según la LOPD especifique.

El grado de seguridad que se le ha asignado a este TFG es el grado 2 puesto que el autor hace un estudio de la legislación aplicable al sistema y el cifrado de los datos.

3.2. Resultados

En este apartado se discuten los resultados obtenidos tras el análisis de los TFG de forma individual. El objetivo es obtener conocimiento sobre el comportamiento de los autores respecto a la protección de credenciales de usuario, es decir, si se ha mejorado o empeorado desde un punto de vista cronológico. Adicionalmente, se comprobará este comportamiento dependiendo del tipo de sistema desarrollado, que puede ser aplicación móvil, aplicación web o programa. También se expondrán las razones posibles del comportamiento de los autores de los TFG para que se hayan obtenido los resultados que se mostrarán con el apoyo de gráficas. Por último, se comentarán las estrategias que se han observado a la hora de abordar la seguridad cuando se desarrolla el proyecto.

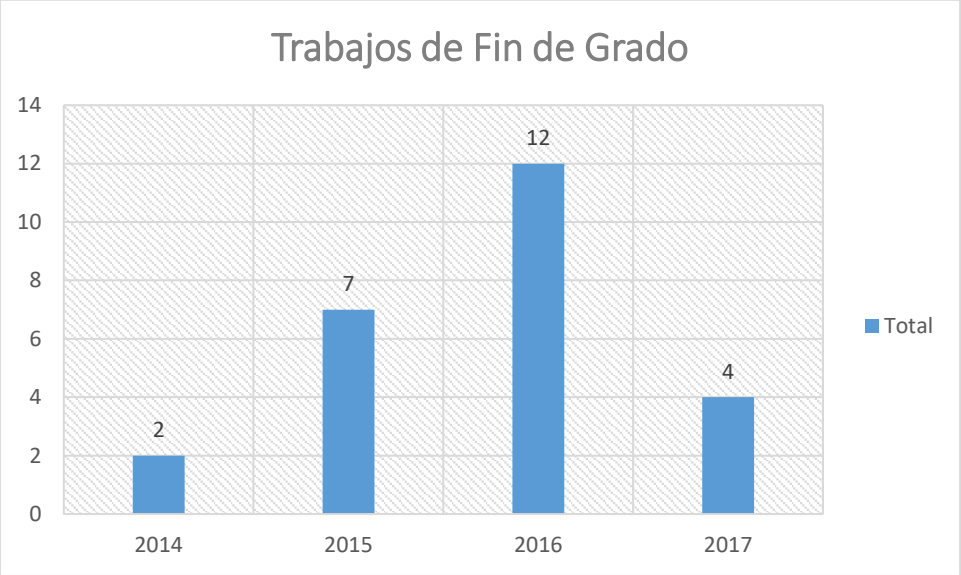


Figura 14: Número de TFG ordenados cronológicamente

En este gráfico se puede apreciar el número de trabajos de fin de grado dependiendo de cuando se defendieron, es decir, de qué año datan los TFG escogidos como muestra para el análisis. El rango que se puede observar es de tres años, con otro TFG de hace cuatro años. Es relevante mencionar que el tener en cuenta que el periodo de tres a cuatro años atrás posiciona al estudio desde 2014 a 2017. Durante estos cuatro años de diferencia cabe remarcar que se ha intentado incrementar la seguridad informática debido a diversos sucesos de fallos de seguridad en sistemas y usurpación de credenciales de usuario y datos personales de empresas, como es el caso de Facebook. Adicionalmente, esto posiciona al estudio en el rango de tiempo en el que tras la PHC (*Password Hashing Competition*) se conoció la nueva función hash ganadora e idealmente más segura hasta el momento Argon2. También hay que tener en cuenta que durante este periodo de tiempo ya existían ataques contra las funciones MD5 y SHA1, entre los que figuran el ataque a los certificados de Microsoft [43] y los ataques SHAppening [44] y SHAttered [10]. Principalmente se debe tener en cuenta que, aunque estas funciones fueran conocidos como algoritmos no seguros, de entre los pocos alumnos que especificaron el algoritmo a emplear en su sistema se escogió el uso de MD5 en dos de las muestras.

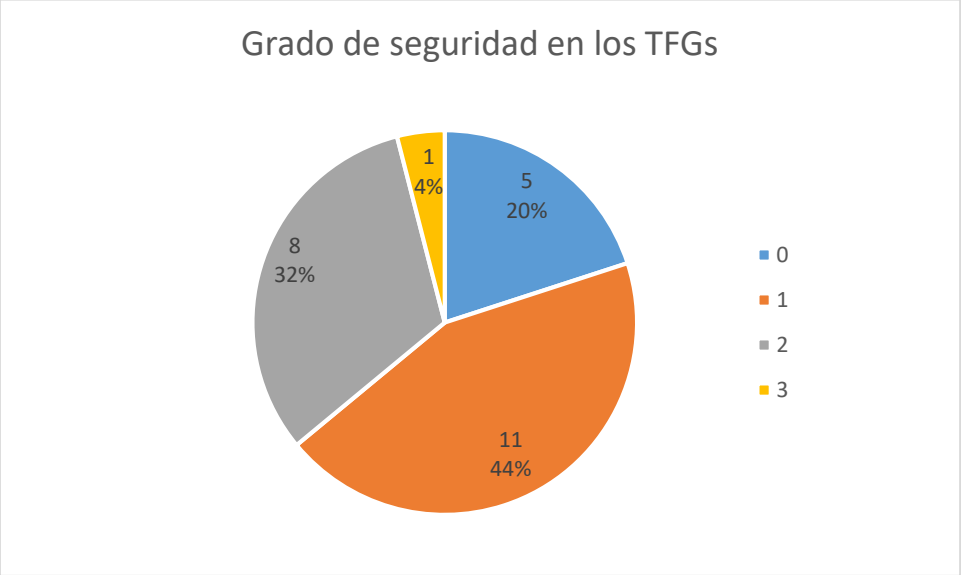


Figura 15: Distribución de grados de seguridad

En este gráfico circular, se puede apreciar la distribución total de los grados de seguridad en las muestras. Como se puede observar en la leyenda los distintos grados se seguridad están asignados a un color correspondiente. Dentro de cada una de las particiones del gráfico circular se puede observar el porcentaje de los TFG que poseen ese grado de seguridad, así como el número total de muestras que representan esos porcentajes. Se puede observar que no existe ningún valor para el grado 4 de seguridad debido a que no se ha adjudicado a ninguna de las muestras analizadas. La escala de seguridad empleada es la misma que se definió al inicio del tercer capítulo y es la siguiente:

- **Grado 0:** este grado es el más bajo de todos y significa que a pesar de que el proyecto trata con datos de usuarios y requiere del uso de las credenciales de usuario, el autor omite completamente la seguridad y no hace requisitos de seguridad ni referencia a las legislaciones aplicables al sistema.
- **Grado 1:** en este grado de seguridad, el autor hace al menos mención o a unos requisitos de seguridad debido al tratamiento de datos de usuarios o se menciona de forma somera la legislación aplicable. Con que se dé una de las dos opciones se obtiene este grado de seguridad.
- **Grado 2:** este grado de seguridad es igual que el Grado 1, con la diferencia que es necesario que se mencionen ambas opciones en el documento del proyecto.
- **Grado 3:** este grado de seguridad amplía el Grado 2 de seguridad, de forma que se espera que se especifique la forma de la que se va a implementar la seguridad mencionada en los requisitos de seguridad o si se ha realizado un análisis en detalle del marco regulador o legal.
- **Grado 4:** este es el último grado y es el que se debería aspirar a tener, en este grado se hace mención de los requisitos de seguridad y se argumenta las decisiones de los mecanismos de seguridad empleados o se especifica qué algoritmo se empleará. Adicionalmente, se hace un análisis en profundidad del marco regulador o marco legal.

El siguiente punto a destacar, es la existencia de un único trabajo de fin de grado con el grado 3 de seguridad, que en realidad es un grado que se obtiene de manera sencilla si se realizan unos requisitos de seguridad en los que se especifica el método de protección de credenciales de usuario, el algoritmo que se va a emplear y un estudio de la legislación aplicable al sistema en lo relativo a las credenciales de usuario que generalmente implica la necesidad de cumplir la LOPD,

ya que la RGPD no se puede evaluar porque entró en vigencia a finales de 2017. A partir del grado 2 de seguridad se trata con el 96% de los TFG analizados, es necesario aclarar que el nivel 2 de seguridad simplemente se obtiene con la mención en el documento del uso de la LOPD y la mención de requisitos de seguridad sin necesidad de especificar el algoritmo a emplear. El cambio entre el grado 2 al grado 3 podría decirse que define un nivel de diferenciación entre autores que se han preocupado en investigar un mínimo para saber qué tipo de algoritmos existen y cuál van a usar en su sistema, teniendo esto en cuenta el 32% de las muestras poseen un grado 2 de seguridad.

En cuanto al grado 1 es más preocupante ya que solamente se necesita hacer mencionar una de las dos siguientes cuestiones para alcanzar este nivel, ya sea una especificación de requisitos de seguridad en la que al menos se menciona la protección de las credenciales de usuario independientemente del método a emplear o haciendo referencia a la LOPD y su aplicación en el sistema desarrollado. Esto quiere decir que un 44% de los autores de los TFG solamente tienen en cuenta una de estas dos cuestiones al desarrollar el proyecto. Es una cifra alarmante ya que con simplemente tener un requisito en el que se especifique que se cifrará la base de datos se puede entrar a esta categoría o con tener un apartado en el que se menciona la aplicación de la LOPD. Pero el problema principal es que sólo se necesita una de las dos opciones, lo cual demuestra un nivel de falta de concienciación o preocupación por el ámbito de la seguridad en el desarrollo de sistemas por parte de los alumnos. Finalmente, se puede observar que un 20% de los trabajos de fin de grado analizados no hacen mención alguna sobre la seguridad de las credenciales de usuario en el sistema que han desarrollado. Es cierto que, este dato no es tan inesperado ya que se suponía desde un principio que un conjunto de los trabajos de fin de grado de los alumnos analizados seguramente no tendría en cuenta la seguridad en el proyecto. La hipótesis que se barajaba inicialmente era una posible falta de seguridad debido al desarrollo de un prototipo, pero en general el sistema desarrollado no era un prototipo, simplemente no se ha tenido en cuenta la seguridad en el proyecto.

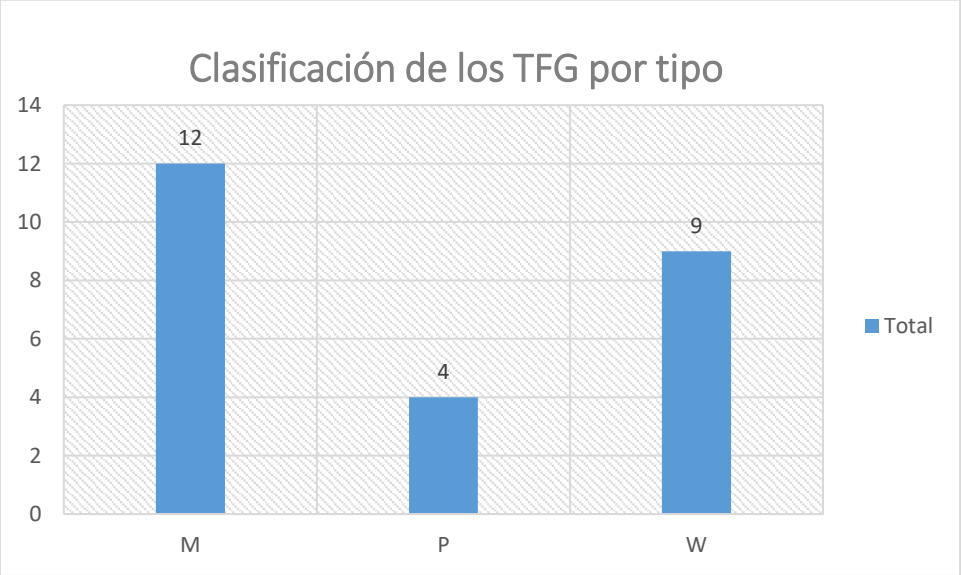


Figura 16: Clasificación de los TFG por tipo de aplicación

En este gráfico se muestran los TFG analizados dependiendo del tipo de proyecto desarrollado, las iniciales corresponden con aplicación móvil (M), programa o sistema (P) y aplicación web (W) respectivamente. Las aplicaciones móviles se refieren a un desarrollo del proyecto para un móvil, generalmente orientados a Android, aunque se encontró un TFG desarrollado en iOS. Los programas o sistema son proyectos que desarrollan una aplicación que generalmente se espera

ejecutar en un computador como puede ser algunas de las simulaciones 3D analizadas. Finalmente, los TFG que son aplicaciones web suelen ser el desarrollo de un *eCommerce* o página web para una organización. Otra característica que destacar es que el desarrollo de aplicaciones móviles para Android y el desarrollo de aplicaciones web suelen emplear el lenguaje Java. Las bases de datos más comúnmente empleadas son la base de datos relacional MySQL y en algunos casos la base de datos no relacional MongoDB.

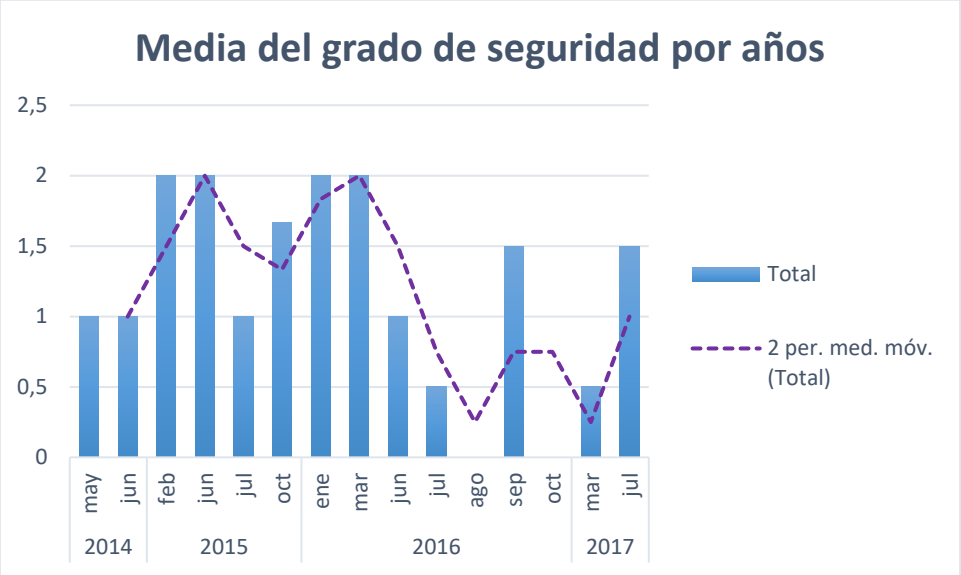


Figura 17:Media del grado de seguridad agrupado por años

En este gráfico se muestra la media de seguridad en función del año en el que se realizó la defensa del trabajo de fin de grado. Se puede observar la variación en los valores de seguridad según pasan los años, adicionalmente se ha añadido las barras de error representando la desviación típica de los valores que está en el rango de 0,5 a 1,714. El valor de la media de seguridad empieza en el grado 1 de seguridad en 2014. Luego se puede observar un aumento de la media de seguridad a lo largo de 2015 y 2016. Entre las posibles razones pueden estar el aumento de la importancia de la seguridad informática debido a que las criptomonedas empezaron a cobrar gran importancia el mercado y el método para generarlas es mediante mecanismos criptográficos como SHA-256 o *scrypt*. A finales de 2016 hay un descenso del grado de seguridad al que no se le ha encontrado ninguna causa posible. Una de las hipótesis es una diferenciación entre dos tipos de alumnos en el año 2016, los que eran conscientes de la necesidad de la seguridad y los que no. Adicionalmente, es posible que los TFG defendidos a principios de 2016 se correspondan con alumnos que podrían haber acabado los estudios en 2015, pero terminaron presentando en la primera convocatoria de 2016. De esta forma es más factible el hecho de que existan dos grupos diferenciados de alumnos que tienen en cuenta la seguridad y otros que no.

Continuando con el análisis cronológico, a inicios del año 2017 se vuelve a ver un aumento en la media del grado de seguridad que puede deberse a la aparición de varios ataques y filtrado de datos personales. Hay que tener en cuenta que un alumno que presenta el TFG en 2015, suponiendo que supere todas las asignaturas de cada año, ha recibido los conocimientos sobre seguridad en los 2 años previos. Teniendo este supuesto en cuenta la explicación sobre las criptomonedas tiene un mayor sentido ya que el ascenso del *Bitcoin* empezó en 2013 y posiblemente los conceptos relacionados con la seguridad informática fueran más atractivos a los alumnos. Por tanto, estos valores de la media del grado de seguridad representan el grado de seguridad de los alumnos que presentaron el TFG ese año, pero no la concienciación actual

sobre la seguridad durante ese año. En definitiva, se ha mostrado una vuelta al valor inicial de la media del grado de seguridad a pesar de la subida presente en 2015 y principios de 2016.

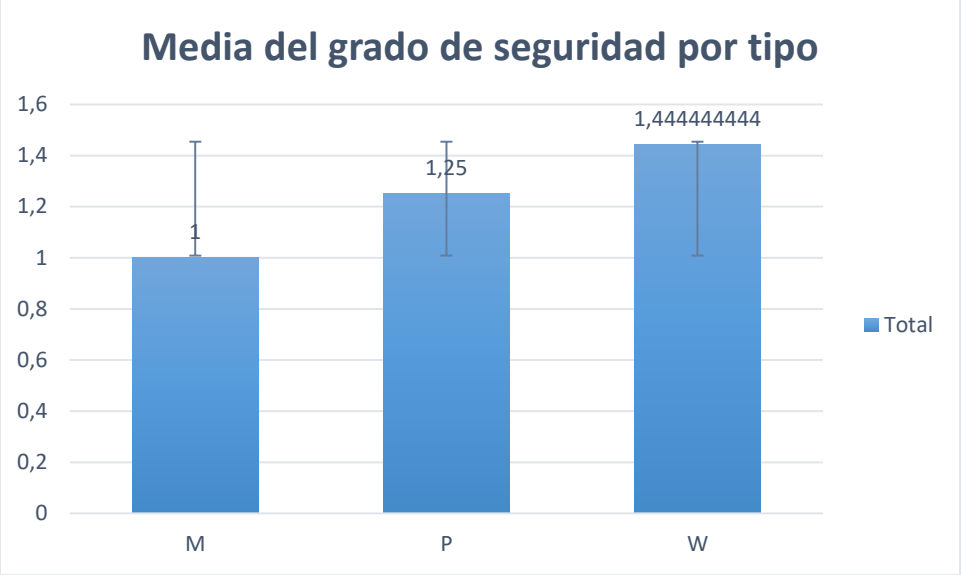


Figura 18: Media del grado de seguridad por tipo de aplicación

En este gráfico se puede apreciar la media general del grado de seguridad dependiendo del tipo de aplicación que se evalúa. De entre los tres tipos de aplicación definidos el grado de seguridad en las aplicaciones móviles es el más débil con un valor medio de 1. El tipo de aplicación que aporta un mayor grado de seguridad son las aplicaciones web con un valor de 1,4 periodo. Este comportamiento puede deberse a que generalmente las aplicaciones web requieren de un manejo de base de datos y al estar relacionadas con sistemas *eCommerce* los datos almacenados son bastante críticos. En cambio, las aplicaciones móviles desarrolladas están orientadas al *e-learning* y aplicaciones de carácter social, por tanto, los autores no deben haber tenido en cuenta demasiado la seguridad, aunque deberían haberla tenido en cuenta.

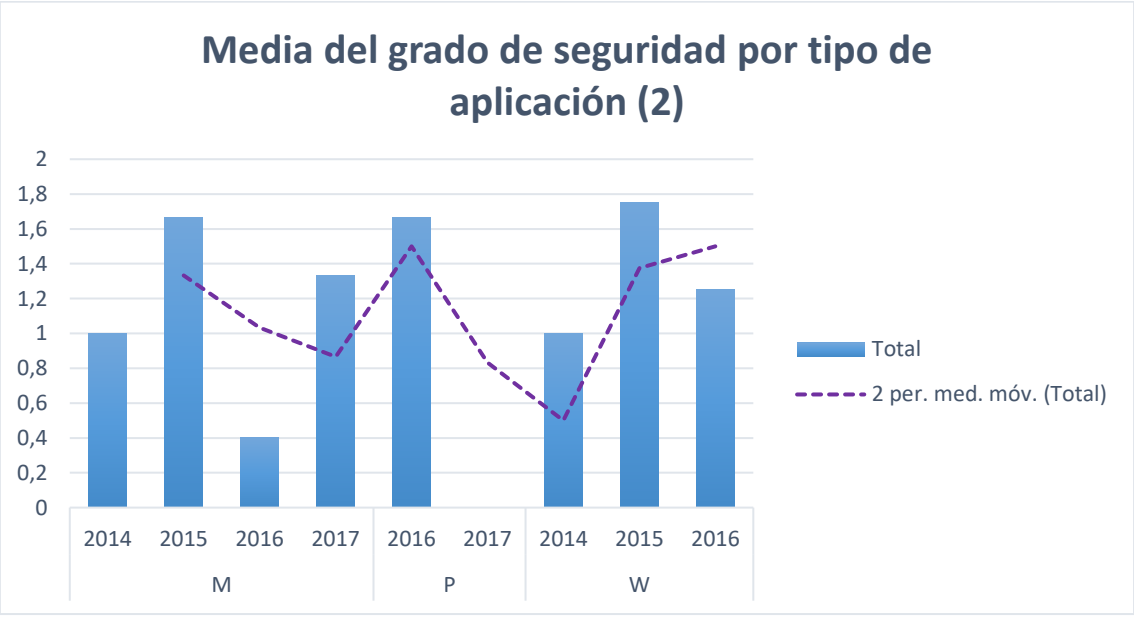


Figura 19: Media del grado de seguridad por tipo de aplicación dependiendo del año

En este gráfico se muestra el grado medio de seguridad agrupándolo por tipo de aplicación y año. En base a este gráfico se puede observar que el grado de seguridad en aplicaciones móviles sigue el patrón que se encontró en la gráfica previa donde se ve un aumento desde 2014 hasta

2015 pero en 2016 se ve un descenso considerable y por último se vuelve a un grado aproximadamente igual al grado de seguridad que había en 2014. En el caso de las aplicaciones o programas, se aprecia un descenso entre los dos únicos años que existen, esto se debe a que solamente hay una muestra en 2017 y está valorada con un grado de seguridad 0. Por último, las aplicaciones web demuestran un crecimiento final respecto al grado de seguridad inicial. Para comprender en mayor detalle cual es la causa de la bajada del grado de seguridad desde 2015 a 2016 se empleará el siguiente gráfico.

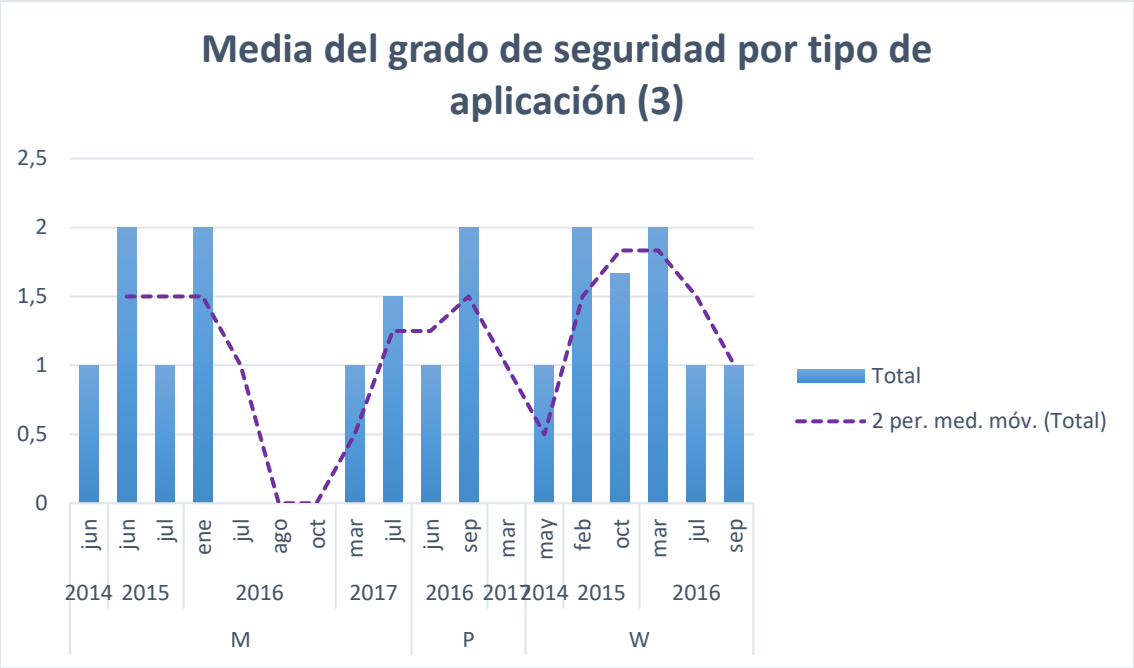


Figura 20: Media del grado de seguridad por tipo de aplicación dependiendo del mes

En este gráfico se puede apreciar el desglose del gráfico anterior con mayor precisión respecto al año. Aquí es donde se muestra la bajada súbita del grado de seguridad entre 2015 y 2016, en específico durante los meses de julio, agosto y octubre. Gracias a este gráfico se puede comprender el porqué de la bajada del grado de seguridad que se estudiaba en el gráfico con la media de seguridad por años. El descenso repentino se debe a un conjunto de aplicaciones móviles desarrolladas que se defendieron entre julio y octubre que no tuvieron en cuenta el grado de seguridad, debido a que el valor es de 0 afectó de forma impactante a la otra gráfica y demostraba ese comportamiento que en principio no tenía explicación. En cuanto a las aplicaciones o programas, se ve un aumento en el año 2016 para luego bajar hasta el grado 0 en 2017. De todos los tipos de aplicaciones las aplicaciones web muestran los mejores resultados ya que empiezan con un grado 1, luego toman un impulso hasta el grado 2 y por último vuelven a mantenerse en el grado 1.

Se puede concluir entonces que de forma general se empezó el año 2014 con un grado 1 de seguridad que tuvo una subida en 2015 y principios de 2016 para luego bajar a finales de 2016 y 2017 al valor previo en 2014. De todos los tipos de aplicaciones desarrolladas, se puede asegurar que las aplicaciones web son aquellas que han tenido más en cuenta la protección de las credenciales de usuario, en principio por que las aplicaciones solían ser portales *eCommerce* y los datos tratados son críticos.

Otra de las cosas que hay que destacar es las estrategias que se han observado a la hora de llevar a cabo los TFG. Las estrategias en cuestión hacen referencia a cómo enfrentarse al problema de la seguridad cuando se desarrolla del proyecto. Una de las estrategias es optar por desarrollar una aplicación o sistema que no requiera de autentificación y así evitar el tratamiento de los

datos de los usuarios. Otra estrategia similar se basa en borrar todos los datos generados cuando se ha empleado el sistema, esta estrategia es común en sistemas de simulación. La siguiente no es una estrategia, sino un defecto de cómo abordar el TFG. Tras la lectura de los TFG se ha podido apreciar que muchos autores no le dan la importancia necesaria a la seguridad porque parece que creen que al ser un TFG es un desarrollo de un sistema que no saldrá a un entorno real y se prefieren centrar en aportan tantas funcionalidades como sea posible con el objetivo de presentar un proyecto más atractivo. Finalmente, existe un grupo de autores que son conscientes de que tienen que desarrollar un sistema con autentificación y por tanto tratamiento de datos de los usuarios. El problema es que a pesar de saber que tienen que proveer mecanismos de seguridad para proteger la confidencialidad de los datos, no saben cómo llevarlos a cabo y generalmente deriva en requisitos del estilo “se cifrará la base de datos” y estudios escuetos de la LOPD como “se proveerá al sistema con los mecanismos de seguridad para cumplir con la LOPD”.

4. DISEÑO DE MATERIALES DIDÁCTICOS

4.1. Proceso de diseño y elección de los Materiales Didácticos.

En base a los resultados obtenidos del análisis de los TFG y las soluciones propuestas se procede a realizar un proceso de elaboración y elección de los materiales didácticos. Con este objetivo en mente se debe tener en cuenta que de los alumnos analizados los problemas principales por los que no incluían mecanismos para proteger las credenciales de usuario son:

- No son conscientes que son datos sensibles y deben protegerlos, en parte puede ser porque no conozcan la facilidad con la que se pueden realizar los ataques.
- Sean conscientes de que tienen que asegurar la seguridad de los datos de los usuarios y sólo se les ocurre cifrar la base de datos porque no conocen otros métodos.

Otro punto que hay que tener en cuenta son las soluciones propuestas para asegurar la protección de las credenciales de usuario que se aportaron en el segundo capítulo. Las soluciones propuestas son las siguientes:

- Añadir una política de contraseña.
- Usar funciones hash.
- Añadir un *salt* criptográficamente fuerte.
- Cifrado de datos argumentando el motivo de su uso, no es aceptable el cifrar como opción por defecto.
- Limitar los permisos y responsabilidades de las entidades que tratan con los datos de los usuarios.

4.1.1. Material Didáctico 1

Teniendo en cuenta estos dos conjuntos de ideas, se pensó que uno de los objetivos de los materiales didácticos sería atraer la atención de los lectores para que se dieran cuenta de los ataques que se pueden realizar y lo sencillos que pueden llegar a ser. Con esto en mente se diseñó el primer material didáctico con el que se espera se atraiga la atención del lector, por lo que debe ser entretenido, y que se presenten unos conceptos básicos sobre el tratamiento a aplicar sobre las contraseñas. Para el primer material didáctico se empleará la herramienta *John the Ripper* para romper hashes que sean débiles, de esta manera el lector aprenderá una de las herramientas con las que pueden sufrir ataques y lo fácil que es realizarlo sobre las funciones hash que no son seguras. Además, al usar *John the Ripper*, se usará su versión gratuita para enfatizar en la facilidad de realizar los ataques. Adicionalmente, se empleará una máquina virtual con Kali Linux, que viene por defecto con un conjunto de herramientas que pueden servir potencialmente para atacar sistemas, aunque en el material didáctico no se verá el uso de estas herramientas.

4.1.2. Material Didáctico 2

Una vez en lector haya realizado idealmente el primer material didáctico, al tener su atención captada y un aumento en su nivel de concienciación respecto a la seguridad de las credenciales de usuario, se procede a explicar un método para proteger las credenciales de usuario que no involucre su tratamiento de forma directa. Para ello se introduce una cuestión que servirá al lector en futuras decisiones de diseño en las que se deba tener en cuenta la seguridad. El segundo material didáctico planteará la limitación de los permisos y las responsabilidades de aquellas entidades que traten los datos de los usuarios. Para ello se hace un enfoque al uso de

las bases de datos MySQL y sistemas Unix. En el primer caso se propondrá la creación de otro usuario que no sea el usuario por defecto de MySQL para acceder a la base de datos, adicionalmente se puede aumentar el nivel de la limitación creando un segundo usuario que solamente tenga acceso a las tablas con los datos de los usuarios. Por otra parte, se explicará el sistema de permisos de los archivos en sistemas Unix de forma que el usuario pueda generar un usuario y otorgarle solamente los permisos necesarios para el manejo del sistema.

4.1.3. Material Didáctico 3

El tercero de los materiales didácticos se ha orientado al concepto que se ha intentado transmitir con mayor ímpetu a lo largo del documento, el uso correcto de funciones hash. Para ello, se demostrará primero una implementación sencilla de funciones hash a las que después se le introducirá el concepto del *salt* criptográfico para aumentar la seguridad frente a ataques conocidos contra las funciones hash. Por último, se introducirá una implementación de la función ganadora de la PHC (*Password Hashing Competition*) que es la función Argon2. En este caso se llevará a cabo la implementación en Java debido a su uso extendido en los TFG analizados para desarrollar el sistema.

4.1.4. Material Didáctico 4

En el cuarto de los materiales didácticos se espera proveer de los conocimientos necesarios sobre los tipos de cifrado para que los alumnos sean capaces de discernir y argumentar cuando deben usar cada uno de los tipos de cifrado. Adicionalmente, se hace una breve revisión a los modos de operación de cifradores de bloque que es algo que muchos alumnos no recuerdan. Para este ejercicio, se desarrollará la implementación del cifrado de datos en Android Studio debido a que la mayoría de las aplicaciones móviles están desarrolladas en Android, además de usar las clases en Java lo que permite su implementación también en las aplicaciones web que en las que se ha notado un gran uso de proyectos implementados en Java. En el material didáctico se mostrará tanto el cifrado simétrico como el asimétrico, pero no se proveerá un ejemplo de cifrado híbrido, ya que es trivial hacerlo una vez se conoce como cifrar y descifrar los datos mediante cifradores simétricos y asimétricos.

4.1.5. Material Didáctico 5

El último de los materiales didácticos tiene el objetivo de abordar otro acercamiento a la protección de las credenciales de usuario de forma indirecta. En este ejercicio se expondrá el uso de las expresiones regulares para generar en Java ejemplos de *whitelists* y *blacklists*. De esta forma se le proveerá al usuario una forma de filtrar los datos que recibe el sistema para saber si son válidos o mal intencionados.

4.1.6. Resumen de los materiales didácticos

En este apartado se comprueba que los materiales didácticos planteados cumplan con los puntos expuestos al inicio del apartado, para ello se empleará la siguiente tabla.

TABLA 2: RESUMEN DE LOS MATERIALES DIDÁCTICOS

Material Didáctico	Solución cubierta	Motivación	Forma de protección
Material Didáctico 1	Política de contraseña y funciones hash.	Los alumnos no conocen la necesidad de proteger las credenciales de usuario	Protección de las credenciales de forma directa
Material Didáctico 2	Limitación de permisos y responsabilidades	Los alumnos no saben cómo proteger las credenciales de usuario	Protección de las credenciales de forma indirecta
Material Didáctico 3	Funciones hash y *salt* criptográficamente fuerte	Los alumnos no saben cómo proteger las credenciales de usuario	Protección de las credenciales de forma directa
Material Didáctico 4	Cifrado de datos y justificación de la elección	Los alumnos no saben cómo proteger las credenciales de usuario	Protección de las credenciales de forma directa
Material Didáctico 5	Política de contraseñas	Los alumnos no saben cómo proteger las credenciales de usuario	Protección de las credenciales de forma indirecta

En la tabla se ha expuesto cada uno de los materiales didácticos que se van a desarrollar teniendo en cuenta qué soluciones cubre de aquellas que se han propuesto en el documento. También, se aporta el motivo por el que se ha realizado la elección de ese material didáctico, así como la forma en la que se protege la credencial de usuario que puede ser de forma directa dándole un tratamiento a los datos en cuestión o de forma indirecta empleando métodos para reducir las posibilidades de ataques y el impacto que puedan tener sobre el sistema.

4.2. Materiales Didácticos.

En este apartado se exponen la serie de materiales didácticos desarrollados para que los lectores y en especial los alumnos de la UC3M puedan comprender mejor los conceptos explicados a lo largo del documento.

Cada uno de los materiales didácticos es independiente y puede llevarse a cabo sin la lectura de los otros, aunque no es recomendable. Además, cada uno contendrá una explicación breve teórica de los conceptos que se van a emplear en él. Adicionalmente, la estructura del material didáctico está pensada para que el seguimiento de la actividad sea leve y sencillo para el lector con las correspondientes explicaciones de lo que ocurre en cada uno de los pasos realizados.

El código de los materiales didácticos se puede encontrar en https://github.com/elgranlute/TFG_Proteccion_Credenciales_Usuario.

4.2.1. Material Didáctico 1. Introducción a las funciones hash y sus ataques.

En este apartado se tratarán las funciones hash y los posibles ataques a los que son susceptibles. El objetivo de realizar este ejercicio es dar una primera toma de contacto al usuario con el uso de funciones hash y una breve demostración de una herramienta que puede atacar a los hashes llamada *John the Ripper* [45] para concienciar a los lectores que no es tan difícil realizar un ataque sobre valores hash con un ordenador común, es decir, que no es necesaria una gran inversión en hardware para poder llevar a cabo ataques.

En primer lugar, se hace una breve explicación teórica de lo que es una función hash y algunas de las propiedades que tiene. Una función hash es una función matemática al que, dado un input o mensaje, independientemente de su longitud, genera un output que es único y de tamaño fijo, generalmente llamado hash, hash *digest*, *digest* o *hashed value*. Las funciones hash deben cumplir con las siguientes propiedades para ser consideradas seguras ante ataques:

- **Resistencia a pre-imágenes:** dado un valor de hash *h* debería ser difícil encontrar otro mensaje o input *m* de forma que *h = hash(m)*. Esta propiedad hace referencia a la cualidad de una función de una sola dirección, de forma que teniendo solamente el hash resultante es imposible generar el input del que procede.
- **Resistencia a segunda pre-imagen:** dado un input m_1, debería ser difícil encontrar otro input distinto m_2 de tal forma que hash(m_1) = hash(m_2). En el caso de las credenciales de usuario, se refiere a que, si se obtiene el hash de la contraseña, un input incorrecto que no sea la contraseña no debe generar el mismo hash.
- **Resistencia a colisiones:** debería ser suficientemente difícil encontrar dos mensajes m_1 y m_2, siendo $m_1 \neq m_2$, tal que hash(m_1) = hash(m_2). Este par, en caso de encontrarse, se llama una colisión.

Las funciones hash pueden tener distintos objetivos, entre los más comunes es que sirvan como un código identificativo para realizar búsquedas, como código identificativo para comprobar la integridad de los archivos y también para no almacenar las contraseñas en claro. Por ello, no todas las funciones hash estaban diseñadas en un principio para almacenar contraseñas. Uno de los problemas principales recae en la pobre elección de una función hash criptográficamente fuerte por parte de los desarrolladores al no tener conocimientos suficientes de seguridad informática al llevar a cabo un proyecto.

Otro de los problemas recae en que la mayoría de las contraseñas son similares, o a más de una persona se le ha ocurrido, por tanto, son susceptibles a distintos ataques. Adicionalmente, hay conjuntos de contraseñas que ya han sido comprometidas y aun así dentro de ese conjunto pueden encontrarse contraseñas que un usuario utilice actualmente. Entre los posibles ataques que pueden sufrir las credenciales de usuario están presentes:

- Ataque de fuerza bruta, que intenta probar todas las combinaciones posibles.
- Ataque de diccionario, que reduce el conjunto de las posibles contraseñas dependiendo de unas reglas y un *wordlist*.
- Ataque mediante una *lookup table,* que es una técnica de TMTO (*Time-Memory tradeoff*) para consumir menos recursos de CPU.
- Ataque mediante una *rainbow table,* que es una técnica posterior a las *lookup table* que también se basa en TMTO.

Todos estos ataques menos el de fuerza bruta, explotan la debilidad del factor humano en las contraseñas, ya que como se ha mencionado son similares, repetidas o con tener información personal del usuario fácilmente deducibles.

A continuación, se realizará una pequeña demostración del uso de una herramienta en su versión gratuita llamada John the Ripper para obtener los valores que generaron un conjunto de hashes. Con el objetivo de demostrar la sencillez de ataque sobre las funciones hash que se ha explicado que son vulnerables, se utilizarán valores obtenidos de las funciones MD5 y SHA1.

Para la realización de este ejercicio, se usará el sistema operativo Kali Linux [46] en una máquina virtual empleando VMware [47]. La descarga de la imagen empleada se ha realizado desde la sección de descargas de la página oficial, específicamente se ha empleado la imagen con el nombre de "*Kali Linux Vm 64 Bit 7z*" [48]. Hay que mencionar que la imagen de Kali Linux viene con el programa John the Ripper, pero esta es una versión simplificada y genérica. Por ello, se procederá a explicar cómo instalar John the Ripper para que sea más rápido y posea un mayor número de posibles funciones hash con las que trabajar.

El primer paso es comprobar que la imagen de Kali Linux esté completamente actualizada, por ello se ejecuta el comando `apt-get update`.

A continuación, se instalarán algunas librerías que son prerrequisitos para el correcto funcionamiento de John the Ripper, se emplearán los siguientes comandos:

Instalación de librerías

```
apt-get install build-essential libssl-dev yasm libgmp-dev libpcap-dev
libnss3-dev libkrb5-dev pkg-config libopenmpi-dev openmpi-bin libbz2-
dev
```

Instalación de rexgen [49]

```
cd /opt

git clone https://github.com/magnumripper/JohnTheRipper.git
```

Instalación de John the Ripper [50]

```
apt-get install flex cmake bison git

cd /opt

git clone https://github.com/teeshop/rexgen.git

cd rexgen

./install.sh

ldconfig
```

Configuración de John the Ripper

```
cd JohnTheRipper/src/

./configure --enable-mpi
```

Compilación de John the Ripper

```
make -s clean && make -sj4
```

Test de John the Ripper

```
cd ../run

./john --test
```

El test de John the Ripper puede detenerse empleando Ctrl+C.

El siguiente paso es el uso de la herramienta para atacar algunos hashes, para ello primero se tomarán algunas de las contraseñas más usadas y se generará su correspondiente valor hash que se han obtenido de la lista de contraseñas "*500-worst-passwords.txt*" de la página *Skullsecurity* [6]. Las funciones hash a emplear son MD5 y SHA1, para mayor facilidad se emplearán unos scripts para generar los hashes. El código de los scripts es el que sigue:

Código de script_md5.sh

```
#!/bin/bash
echo -n pass | md5sum | awk '{print $1}'> md5hashes.txt
echo -n 123456 | md5sum | awk '{print $1}'>> md5hashes.txt
echo -n pussy | md5sum | awk '{print $1}'>> md5hashes.txt
echo -n dragon | md5sum | awk '{print $1}'>> md5hashes.txt
echo -n qwerty | md5sum | awk '{print $1}'>> md5hashes.txt
echo -n 696969 | md5sum | awk '{print $1}'>> md5hashes.txt
echo -n shadow | md5sum | awk '{print $1}'>> md5hashes.txt
echo -n abc123 | md5sum | awk '{print $1}'>> md5hashes.txt
echo -n pass | md5sum | awk '{print $1}'>> md5hashes.txt
echo -n fuckme | md5sum | awk '{print $1}'>> md5hashes.txt
echo -n test | md5sum | awk '{print $1}'>> md5hashes.txt
echo -n trustno1 | md5sum | awk '{print $1}'>> md5hashes.txt
echo -n access | md5sum | awk '{print $1}'>> md5hashes.txt
echo -n soccer | md5sum | awk '{print $1}'>> md5hashes.txt
echo -n superman | md5sum | awk '{print $1}'>> md5hashes.txt
echo -n asshole | md5sum | awk '{print $1}'>> md5hashes.txt
echo -n fuckyou | md5sum | awk '{print $1}'>> md5hashes.txt
echo -n jessica | md5sum | awk '{print $1}'>> md5hashes.txt
echo -n wizard | md5sum | awk '{print $1}'>> md5hashes.txt
echo -n biteme | md5sum | awk '{print $1}'>> md5hashes.txt
```

Código de script_sha1.sh

```
#!/bin/bash
echo -n pass | sha1sum | awk '{print $1}'> sha1hashes.txt
echo -n 123456 | sha1sum | awk '{print $1}'>> sha1hashes.txt
echo -n pussy | sha1sum | awk '{print $1}'>> sha1hashes.txt
echo -n dragon | sha1sum | awk '{print $1}'>> sha1hashes.txt
echo -n qwerty | sha1sum | awk '{print $1}'>> sha1hashes.txt
echo -n 696969 | sha1sum | awk '{print $1}'>> sha1hashes.txt
echo -n shadow | sha1sum | awk '{print $1}'>> sha1hashes.txt
echo -n abc123 | sha1sum | awk '{print $1}'>> sha1hashes.txt
echo -n pass | sha1sum | awk '{print $1}'>> sha1hashes.txt
echo -n fuckme | sha1sum | awk '{print $1}'>> sha1hashes.txt
echo -n test | sha1sum | awk '{print $1}'>> sha1hashes.txt
echo -n trustno1 | sha1sum | awk '{print $1}'>> sha1hashes.txt
echo -n access | sha1sum | awk '{print $1}'>> sha1hashes.txt
echo -n soccer | sha1sum | awk '{print $1}'>> sha1hashes.txt
echo -n superman | sha1sum | awk '{print $1}'>> sha1hashes.txt
echo -n asshole | sha1sum | awk '{print $1}'>> sha1hashes.txt
echo -n fuckyou | sha1sum | awk '{print $1}'>> sha1hashes.txt
echo -n jessica | sha1sum | awk '{print $1}'>> sha1hashes.txt
echo -n wizard | sha1sum | awk '{print $1}'>> sha1hashes.txt
echo -n biteme | sha1sum | awk '{print $1}'>> sha1hashes.txt
```

La herramienta *John the Ripper* [4] puede emplear tres modos de funcionamiento para obtener los valores que generan los hashes, en este caso las contraseñas. Los modos de funcionamiento se explican de forma extensa en la página de *Openwall* [51]. La siguiente es una breve explicación de cada modo de funcionamiento:

- *Wordlist*, se hace uso de una *wordlist* en la que están las contraseñas posibles.
- *Single Crack*, es el modo con el que se recomienda usar y se emplea un conjunto de reglas para manipular las contraseñas candidatas.
- *Incremental*, es el modo más potente ya que intentará probar todas las combinaciones posibles, pero ello conlleva una penalización en el tiempo que teóricamente sería infinito.

El método que se usará a la largo del ejercicio es el modo de *Wordlist,* ya que se usará el archivo "rockyou.txt" que se encuentra comprimido en un tar en el directorio */usr/share/wordlists*. Con el fin de proveer al lector con el conocimiento de qué podría hacer un atacante para identificar el tipo de hash almacenado en la base de datos se mostrará un ejemplo sencillo de la herramienta *hash-identifier*, que tal y como indica su nombre sirve para analizar qué tipo de función hash pudo haber sido utilizada para un *digest* específico. Para este ejemplo se empleará el resultado de la función hash SHA256 sobre el valor "*password*" que es "5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8".

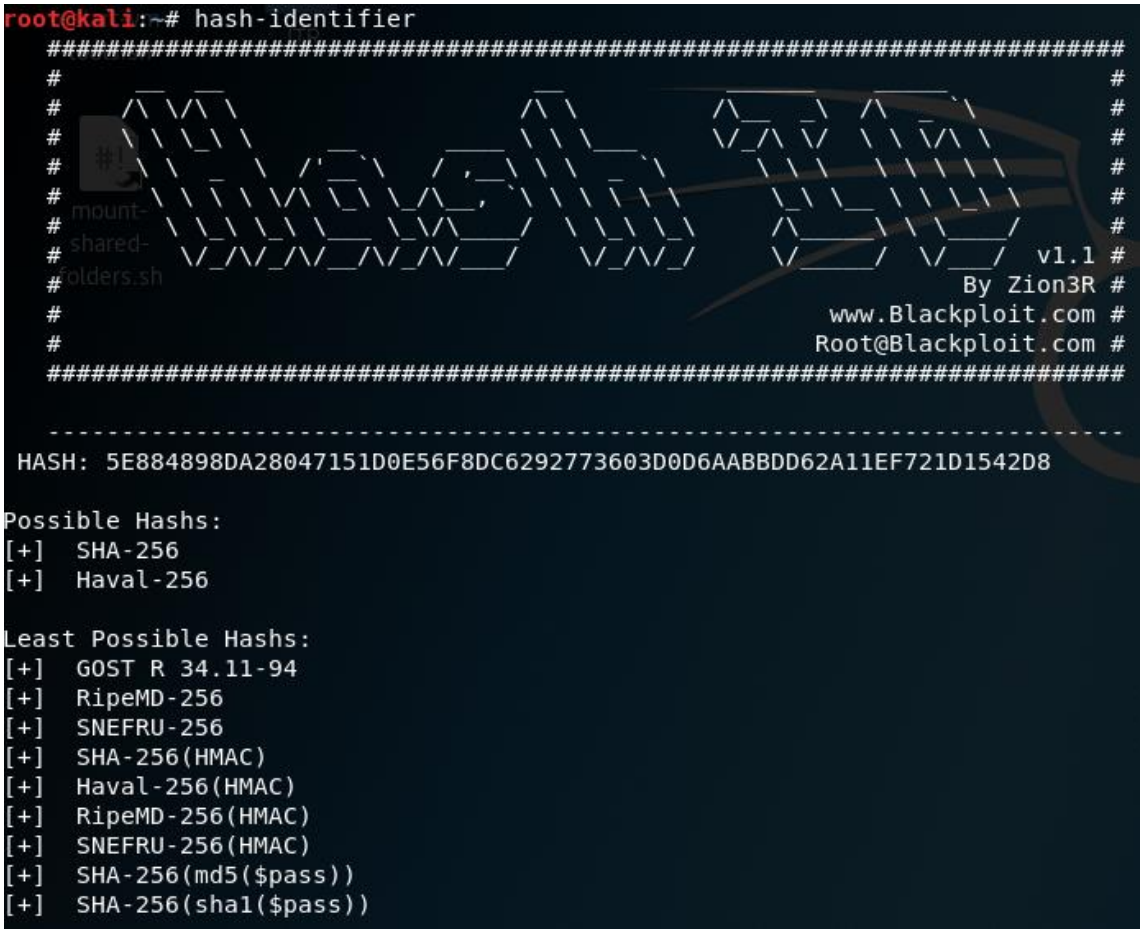


Figura 21: Identificación de un hash SHA-256 usando la herramienta 'hash-identifier'

Tal y como se puede apreciar en la captura la herramienta es capaz de deducir que el *digest* se ha podido generar muy probablemente con SHA256.

El siguiente paso es la ejecución de los scripts previamente definidos, no debe olvidarse dar los permisos para ejecutar el script mediante `chmod a+x <nombre_script.sh>`. El resultado de la ejecución de los scripts es la creación de dos ficheros con los hashes correspondientes.

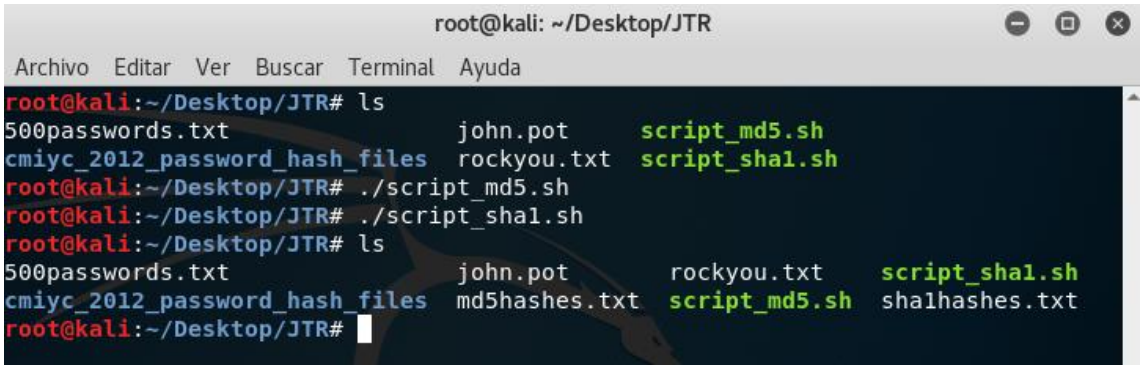


Figura 22: Estado del directorio antes y después de ejecutar los scripts de MD5 y SHA-1

Una vez se tiene el fichero con la *wordlist* y los ficheros con los hashes, se procede a usar John the Ripper para intentar obtener las contraseñas. El primer fichero que se empleará es el que contiene los hashes MD5, luego se empleará el de los hashes SHA1. Para una mayor facilidad en el comando para ejecutar John the Ripper se especificará el formato "Raw-MD5" y "Raw-SHA1" respectivamente, se puede obtener una lista de los formatos disponibles mediante el comando "`./john --list=formats`". El comando empleado para los hashes MD5 es el siguiente:

```
./john --wordlist=/root/Desktop/JTR/rockyou.txt --format=Raw-MD5
/root/Desktop/JTR/md5hashes.txt
```

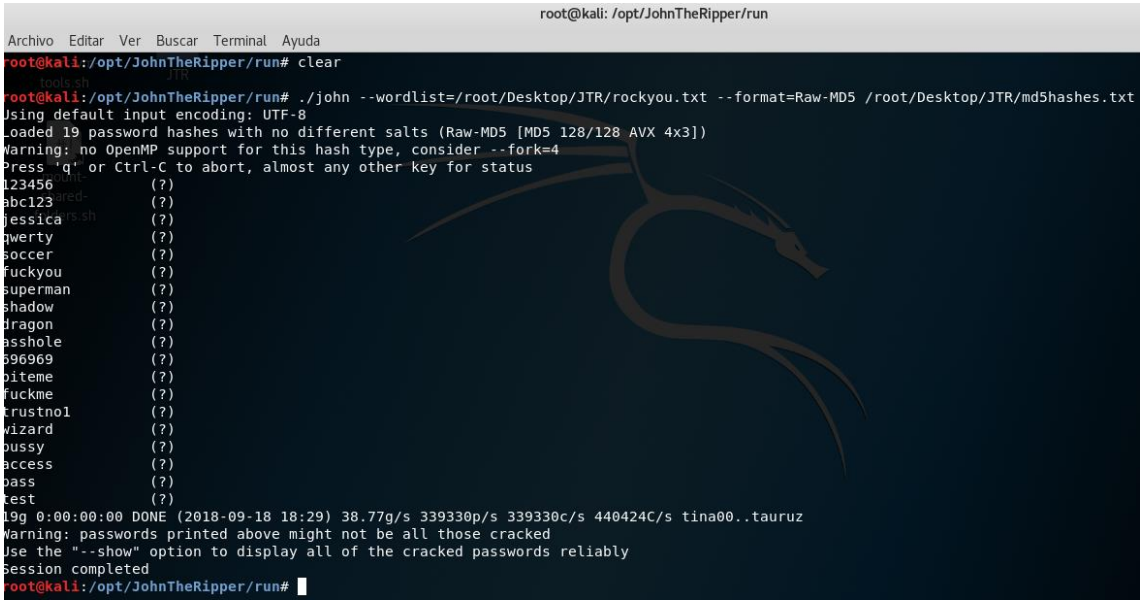


Figura 23: Resultado de crackear la lista de hashes de MD5 con John the Ripper

En el caso de los hashes de SHA1 el comando es el siguiente:

```
./john --wordlist=/root/Desktop/JTR/rockyou.txt --format=Raw-SHA1
/root/Desktop/JTR/sha1hashes.txt
```

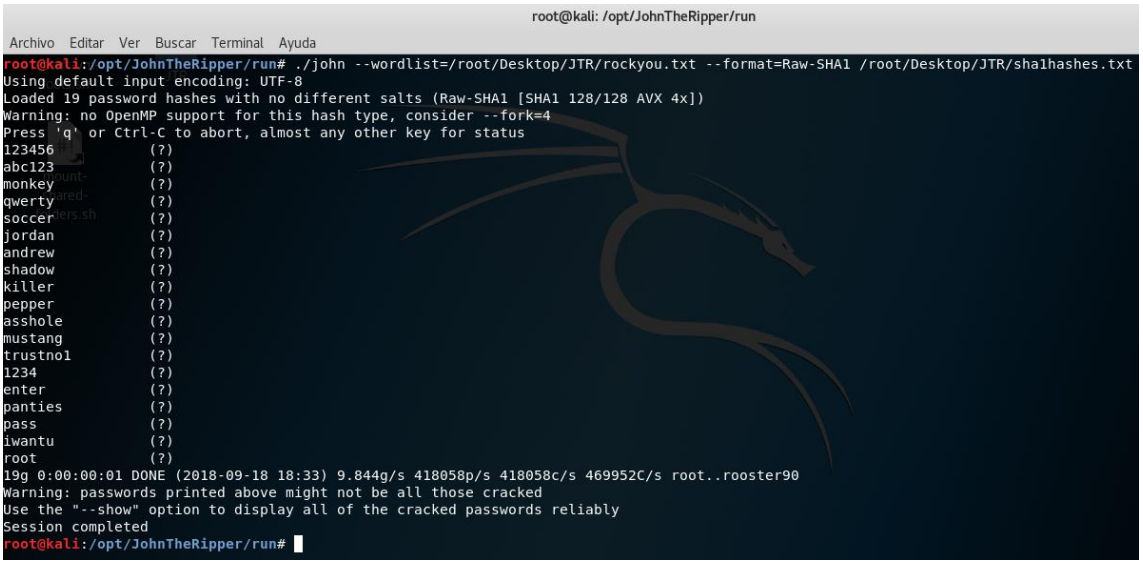


Figura 24: Resultado de crackear la lista de hashes de SHA-1 con John the Ripper

Como se puede observar la mayoría de los hashes han acabado siendo comprometidos y las contraseñas originales son ahora visibles para el atacante.

Por último, hay que añadir que en caso de querer volver a ejecutar el programa con hashes previamente computados la herramienta John the Ripper emplea un archivo llamado *john.pot* en el que se almacenan dichos hashes para no tener que volver a computarlos. Empleando el comando "`./john --show <archivo_hashes>`" se mostrarán los valores a partir de los cuales se generaron los *digest* correspondientes del archivo especificado.

Teniendo en cuenta los resultados obtenidos al emplear la herramienta John the Ripper y *hash-identifier*, se puede deducir que es bastante importante llevar a cabo una correcta protección de las credenciales de usuario almacenadas en el sistema. Para ello, se pueden emplear una serie de técnicas que se describen en el "Material Didáctico 3, uso correcto de funciones hash"

En cuanto a los ataques mencionados de *lookup tables* y *rainbow tables*, son técnicas que permiten que el tiempo de cómputo se reduzca a cambio del uso de memoria cuando se realizan ataques como el que se ha visto en el ejercicio. Actualmente existe un proyecto llevado a cabo por OWASP que permite la generación de *rainbow tables*, llamado OWASP Rainbow Maker Project [52].

4.2.2. Material Didáctico 2. Limitación de responsabilidades y permisos.

En este ejercicio se discutirá la limitación de responsabilidades para el acceso a las bases de datos y los permisos que se le otorgan a los usuarios de un sistema basado en Unix que esté siendo empleado para una aplicación o sistema. El ejercicio se dividirá en dos bloques, el primero se centrará en las bases de datos y el segundo en el manejo de permisos dentro de un sistema Unix.

Generalmente a la hora de realizar las conexiones con la base de datos, la gente emplea el usuario administrador *root* para realizar operaciones. Esto es un error que puede llevar a la pérdida de la base de datos si un atacante se hace con el control de este usuario. Para ello es recomendable la creación de un nuevo usuario a emplear para acceder a la base de datos.

Siguiendo este razonamiento, se debería tener un usuario administrador de la base de datos con todos los privilegios, aunque se debería cambiar la configuración por defecto o típica que suele ser usuario *root* y contraseña *root*. Además, se puede tener un único usuario administrador para cada uno de los esquemas necesitados. El siguiente paso podría ser crear otro usuario y acotar sus privilegios para que tenga acceso únicamente a la tabla que almacene los datos correspondientes a los usuarios registrados en el sistema. De esta forma se tendría un usuario general que es como un super-administrador y un usuario administrador para cada esquema, teniendo en cuenta que normalmente un esquema corresponde con el modelo de datos de una aplicación.

La generación de este usuario se puede realizar con los siguientes comandos. El ejemplo tendrá el supuesto que se tiene la base de datos en local. Primero se muestra el comando para generar el usuario.

CREATE USER '*username*'@'*localhost*' IDENTIFIED BY '*password*';

Los campos *username* y *password* deben sustituirse por los valores correspondientes al nombre del nuevo usuario y la contraseña que empleará. El objetivo es reducir las posibilidades de un ataque sean satisfactorias, por lo que se recomienda evitar contraseñas típicas o emplear el mismo nombre de usuario como contraseña. Es recomendable incluir una mayúscula, minúscula y dígito con una longitud mínima de 8 caracteres. Como alternativa a seguir una política de contraseña, se puede derivar una clave mediante una función de derivación de claves como PBKDF2 y emplear el resultado como contraseña. El siguiente comando sirve para adjudicar los privilegios al nuevo usuario creado.

GRANT ALL PRIVILEGES ON '*database*'.'table' TO '*username'*@'*localhost*';

Con este comando se le adjudicarían todos los privilegios al nuevo usuario sobre la tabla *table* de la base de datos *database*. En el caso de querer limitar aún más los permisos sobre cada tabla se puede especificar de la siguiente forma.

GRANT 'privilege' ON '*database*'.'*table*' TO '*username'*@'*localhost*';

Se puede sustituir el valor *privilege* por SELECT, INSERT, UPDATE, DELETE, EXECUTE y SHOW VIEW que son algunos de los privilegios más asignados. En caso de querer asignar otros privilegios, se puede consultar la lista de privilegios que ofrece MySQL aquí [53]. Por último, para que la asignación de los nuevos privilegios sea efectiva se ejecuta el comando:

FLUSH PRIVILEGES;

En caso de ser más exhaustivo a la hora de restringir los privilegios asignados en la base de datos, se podría definir varios usuarios que tengan acceso a un número de tablas limitado. De esa forma, si uno de los usuarios se ve comprometido, no toda la base de datos quedaría expuesta. En caso de seguir este modelo, se recomendaría hacer al menos un usuario específico para la tabla con los datos de los usuarios registrados en el sistema.

Este segundo bloque del ejercicio se centra en la asignación de permisos a los usuarios del sistema. El concepto es similar al empleado en el bloque anterior, ya que, es conveniente limitar los permisos y las responsabilidades de los usuarios del sistema para que en caso de que un usuario se viera comprometido no afecte a todo el sistema.

En los sistemas Unix los ficheros y directorios pertenecen a un grupo y un usuario. En caso de usar el usuario por defecto se puede acceder a todos los comandos, ficheros y directorios del sistema y eso es peligroso en caso de que dicho usuario se vea comprometido. La recomendación inicial es generar un nuevo usuario y un grupo para asignar permisos sobre los archivos que deba emplear la aplicación a dicho usuario y grupo. Para comprender esto, se explicará brevemente el sistema de permisos de Unix. Cada fichero y directorio tiene una serie de permisos que están divididos en tres secciones, usuario, grupo y el mundo. Cada sección tiene para otorgar tres permisos, que son de lectura, escritura y ejecución. A continuación, se muestra una imagen para mayor claridad.

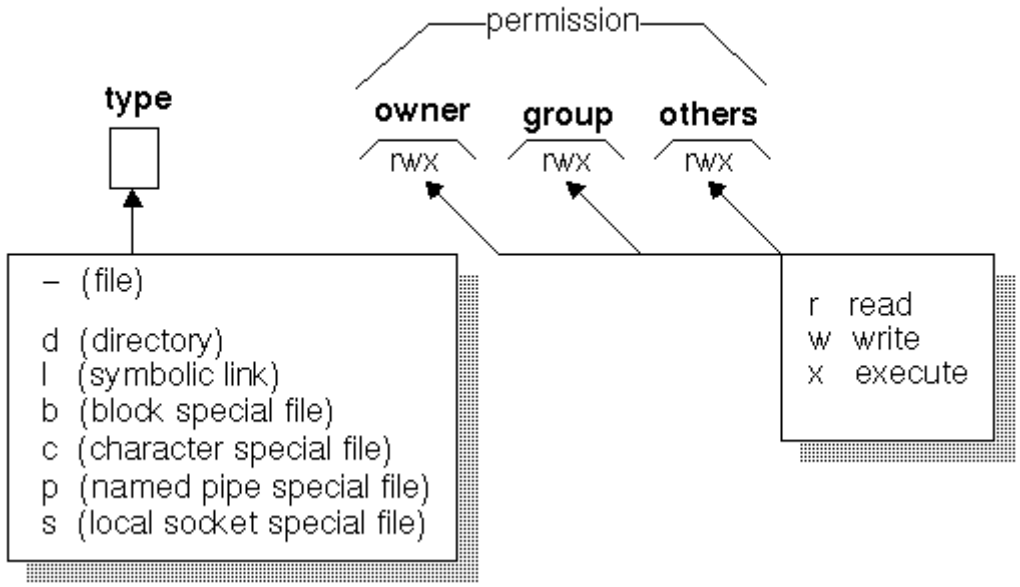


Figura 25: Explicación del sistema de permisos en sistemas UNIX. Figure 5-1 del artículo [74]

Los permisos de lectura, escritura y ejecución se representan con *r, w* y *x* respectivamente. Para la asignación de estos permisos es necesario el uso del comando *chmod XYZ <path>*. Donde X, Y, Z son la representación octal de los permisos adjudicados, por lo que el valor máximo es 7 y el mínimo es 0.

Para la modificación del usuario propietario de los archivos se empleará el comando *chown 'username' <path>.* En caso de que se trate de un directorio se puede emplear la opción *-R* para hacer el comando recursivo y que afecte a todos los archivos dentro del directorio. En caso de querer realizar la modificación del grupo del archivo se emplea el comando *chgrp 'groupname' <path>*. Al igual que antes, si se trata de un directorio se añade la opción *-R*.

Es posible que se quiera comprobar que grupos y usuarios existen en el sistema, para ello se pueden emplear los comandos *sudo cat /etc/passwd* y *sudo cat /etc/group* respectivamente.

Por último, hay que mencionar que una de las estrategias que suele ser más efectiva es seguir una política de bloquear todo y habilitar sólo los permisos necesarios en vez de ir usuario por usuario denegando los permisos que no deberían usar. En caso de seguir la primera política con un nuevo usuario, esto aseguraría que ese usuario sólo tendría acceso a los archivos y directorios estrictamente necesarios. En caso de que no los tenga el error se detectaría rápido porque posiblemente generase complicaciones a la hora de realizar su cometido, ya sea desplegar aplicaciones o mantener el sistema. En cambio, la segunda política es muy susceptible al fallo humano y que se pase por alto algún permiso que no debería estar habilitado.

4.2.3. Material Didáctico 3. Uso correcto de funciones Hash.

En este ejercicio se explicará cómo hacer un uso correcto de las funciones hash con fines criptográficos para la protección de las credenciales de usuario.

Actualmente existen varios problemas relacionados con las contraseñas que emplean los usuarios para identificarse en distintos sistemas. El par de contraseña y usuario conforma la credencial de usuario de una persona física. Las credenciales de usuario son el método mediante el cual un usuario se identifica ante un sistema y la forma más común es el par usuario-contraseña.

El primero de los problemas recae en el poco conocimiento de los usuarios para escoger contraseñas seguras. Esto se debe a que el área de la informática ha avanzado rápidamente en los últimos años y pocos son los usuarios que se mantienen al día con la tecnología. El hecho de tener una contraseña segura es imprescindible hoy en día. La cuestión es que muchos de los usuarios crearán una contraseña que sea fácil de recordar, evidentemente es necesario recordar las contraseñas que se usan.

La forma en la que los usuarios escogen estas contraseñas es la mayor de las debilidades. Una posibilidad es que empleen datos personales o de familiares porque hay muchas probabilidades de que así no se les olvide la contraseña. Otro problema es que los usuarios generalmente emplearan contraseñas dentro de su diccionario natal, es decir, alguien de España usará una contraseña que sea una palabra o combinación de palabras que tengan sentido en el español (castellano). Este es un factor que realmente delimita el conjunto de posibles contraseñas que un usuario puede llegar a utilizar. Es cierto, que existen usuarios que intenten hacer modificaciones a la contraseña que tenían pensada usar con el objetivo de hacerla más difícil de adivinar, pero generalmente son modificaciones simples o permutaciones de caracteres que son conocidas.

Algunos sistemas o aplicaciones lo que hacen es proveer de una política de contraseñas de forma que los usuarios añadan variaciones a la contraseña para que no sea tan evidente. Entre las variaciones más conocidas están:

- Al menos una letra minúscula
- Al menos una letra mayúscula
- Al menos un digito
- La contraseña debe tener una longitud mínima, y a veces se especifica la longitud máxima

Estas políticas de contraseñas, aunque estén bien intencionadas, generalmente no consiguen el resultado esperado debido a que los usuarios harán las modificaciones mínimas a su contraseña original para cumplir los requisitos mínimos de la política de contraseñas. Esto se traduce en que los usuarios cambiarán la primera letra por una mayúscula y añadirán al final de la contraseña un dígito o serie de dígitos evidente como puede ser, 123 o el año de nacimiento del usuario.

Todas estas posibles contraseñas y variaciones de ellas son susceptibles a ataques de diccionario basados en contraseñas que ya han sido comprometidas y mediante el uso de reglas para variar una contraseña.

Primeramente, para evitar que las contraseñas de los usuarios queden comprometidas en caso de que el sistema sea atacado y se filtre la información, se recomienda el uso de funciones hash.

Las funciones hash tienen el objetivo de generar a partir de una contraseña conocida por el usuario un resultado que es único para ese input y servirá como clave del usuario dentro del sistema. El simple uso de las funciones hash sin embargo no es suficiente, ya que, se puede dar el caso que dos usuarios tengan la misma contraseña y aplicando la misma función hash se obtendría el mismo resultado. De esta forma si una de las dos cuentas con la misma contraseña se viera afectada y su contraseña fuera comprometida el atacante sabría también la contraseña de otra cuenta que tiene la misma contraseña. Por tanto, el uso de únicamente las funciones hash evita que el atacante tenga acceso a las contraseñas en claro si el sistema se ve comprometido.

Para mejorar la seguridad de las credenciales de usuario existe una técnica que combate de forma muy eficaz los ataques empleados contra los hashes. Se trata del uso de *salts* criptográficos, los *salts* criptográficos son un conjunto de bytes aleatorios que se generan mediante un CSPRNG (*Cryptographically Secure PseudoRandom Number Generator*). El uso principal del *salt* para las funciones hash es el de añadir entropía, dicho de otra forma, añadir aleatoriedad. El *salt* generalmente se añade al inicio o final de la contraseña del usuario para posteriormente usarlo como input en la función hash que usa el sistema. Con el simple hecho de usar *salts* los ataques a los que antes las funciones hash eran vulnerables, ahora se vuelven cercanos a no ser factibles. Adicionalmente, el caso en el que dos o más usuarios compartan la contraseña deja de ser un problema debido al *salt.* Para cada uno de los usuarios se debe emplear un *salt* distinto. Nunca se debe reciclar el uso de un salt, tampoco se recomienda en caso de permitir modificar la contraseña emplear el *salt* de la contraseña previa.

Una vez se han aclarado los conceptos teóricos, se procede a seguir el ejemplo de uso de las funciones hash en Java.

En Java existen por defecto un conjunto de funciones hash que están incluidas en *java.security.MessageDigest*. Con objetivo de cubrir todas las funciones se mostrará a continuación un ejemplo de las funciones hash que proporciona Java que son:

- MD5
- SHA1
- SHA256

Para la representación de los valores devueltos por los métodos implementados de las funciones hash, se empleará un método propio para poder mostrar el valor en hexadecimal. Dicho método es el siguiente:

```
public static String bytesToHex(byte[] array) {
    StringBuffer hexString = new StringBuffer();
    for (int i = 0; i < array.length; i++) {
    String hex = Integer.toHexString(0xff & array[i]);
    if(hex.length() == 1) hexString.append('0');
        hexString.append(hex);
    }
    return hexString.toString();
}
```

Figura 26: Código del método bytesToHex

La primera de las funciones es MD5, que actualmente no se emplea como función hash criptográfica por ser susceptible a ataques de colisión.

```java
public static String hashWithMD5(String password){
    String hash = null;
    MessageDigest md5;
    byte [] hashedbytes = null;
    try {
        //Obtener MD5
        md5 = MessageDigest.getInstance("MD5");
        //Pasar los bytes del string password
        hashedbytes = md5.digest(password.getBytes(StandardCharsets.UTF_8));
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    if(hashedbytes!=null){
        //transformar el resultado de byte[] a String en hexadecimal
        hash = bytesToHex(hashedbytes);
    }
    return hash;
}
```

Figura 27: Código del método para hashear con MD5

La siguiente función hash es SHA-1 que tampoco se emplea como función hash criptográfica por ser vulnerable a ataques.

```java
public static String hashWithSHA1(String password){
    String hash = null;
    MessageDigest sha1;
    byte [] hashedbytes = null;
    try {
        //Obtener SHA1
        sha1 = MessageDigest.getInstance("SHA-1");
        //Pasar los bytes del string password
        hashedbytes = sha1.digest(password.getBytes(StandardCharsets.UTF_8));
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    if(hashedbytes!=null){
        //transformar el resultado de byte[] a String en hexadecimal
        hash = bytesToHex(hashedbytes);
    }
    return hash;
}
```

Figura 28: Código del método para hashear con SHA-1

El siguiente método emplea la función hash SHA-256. Esta es una función de la familia SHA-2, con la cualidad de que la longitud de salida del hash es de 256 bits. La longitud máxima del input para SHA-256 es de $2^{64} - 1$, dicha longitud medida en bits.

```java
public static String hashWithSHA256(String password){
    String hash = null;
    MessageDigest sha256;
    byte [] hashedbytes = null;
    try {
        //Obtener SHA256
        sha256 = MessageDigest.getInstance("SHA-256");
        //Hashear los bytes del string password
        hashedbytes = sha256.digest(password.getBytes(StandardCharsets.UTF_8));
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    if(hashedbytes!=null){
        //transformar el resultado de byte[] a String en hexadecimal
        hash = bytesToHex(hashedbytes);
    }
    return hash;
}
```

Figura 29: Código del método para hashear con SHA-256

Hasta este punto se ha mostrado como se pueden emplear las funciones hash aportadas por Java. Pero tal y como se ha explicado al inicio del documento el uso de las funciones hash por sí solas no es suficiente. Por ello, el siguiente paso es añadir una forma de generar un *salt* criptográficamente fuerte. En Java se puede emplear el CSPRNG de SecureRandom importando java.security.SecureRandom. A continuación, tal y como se ha hecho con las funciones hash, se mostrarán varios ejemplos de cómo obtener un *salt* en Java.

El primer ejemplo es uno genérico en el que se crea el objeto *SecureRandom* y se emplea el método *nextBytes(byte [] bytes)* para obtener la cantidad de bytes aleatorios especificada. En este caso al no especificar el algoritmo que se quiere emplear para obtener los bytes aleatorios, Java empleará el que tenga por defecto dependiendo del sistema operativo.

```java
/**
 * Metodo para generar un salt a partir de un CSPRNG mediante java.security.SecureRandom
 * Argumentos:
 *  int length: longitud en bytes del salt a generar
 * Valor devuelto:
 *  Un array de bytes que es el salt generado de longitud length*/
public static byte[] generateSalt(int length){
    byte[] salt = new byte[length];
    //Por defecto en Windows suele ser SHA1PRNG
    SecureRandom random= new SecureRandom();
    random.nextBytes(salt);
    return salt;
}
```

Figura 30: Código del método básico para generar un salt

En caso de trabajar en Windows, el valor por defecto de SecureRandom es el algoritmo SHA1PRNG. Debido a que SHA1PRNG puede ser susceptible a la predicción de los valores que se generan, se puede optar por usar el PRNG (*PseudoRandom Number Generator*) que aporta el sistema operativo.

Para el caso de Windows el método queda de la siguiente forma:

```java
//Si se trabaja en Windows y no se quiere emplear SHA1PRNG, se puede optar por usar Windows-PRNG
public static byte[] generateSaltWindows(int length) throws NoSuchAlgorithmException{
    byte[] salt = new byte[length];
    SecureRandom random= SecureRandom.getInstance("Windows-PRNG");
    random.nextBytes(salt);
    return salt;
}
```

Figura 31: Código del método para generar salt usando el PRNG de Windows

En el caso de trabajar con un sistema Unix o Linux el método queda de la siguiente manera:

```java
//Si se trabaja en un sistema UNIX/Linux y no se quiere emplear SHA1PRNG, se puede optar por usar NativePRNG
public static byte[] generateSaltUnix(int length) throws NoSuchAlgorithmException{
    byte[] salt = new byte[length];
    //NativePRNG emplea el directorio /dev/urandom como fuente de entropia
    SecureRandom random= SecureRandom.getInstance("NativePRNG");
    random.nextBytes(salt);
    return salt;
}
```

Figura 32: Código del método para generar salt usando el PRNG Nativo (UNIX)

En el caso del sistema Unix o Linux, existe una discusión por el método para obtener la entropía para el CSPRNG. Esto se debe a que se puede emplear el directorio */dev/random* o el directorio */dev/urandom*. La diferencia más importante es que si se emplea */dev/random* la obtención de la entropía puede conllevar que la llamada sea bloqueante hasta que el sistema decida que existe suficiente entropía verdadera. En el caso de emplear */dev/urandom* las llamadas no son bloqueantes bajo la posibilidad de que los datos no sean realmente aleatorios. Debido al enfoque del uso de estos *salts* para aplicaciones en las que generalmente la respuesta del sistema debe ser rápida, el uso de */dev/urandom* es recomendado. Para una mayor compresión del problema se puede comprobar la siguiente página web [54].

Una vez se ha comprendido y se tiene un método para obtener un *salt* de la longitud deseada, se puede modificar el método de la función hash SHA-256 para incluir el uso del *salt*.

```java
public static String hashWithSHA256Salted(String password, byte[] salt){
    String hash = null;
    MessageDigest sha256;
    byte [] hashedbytes = null;
    try {
        //Obtener SHA256
        sha256 = MessageDigest.getInstance("SHA-256");
        //Concatenar los bytes de salt al final de los bytes de la password
        byte [] passBytes = password.getBytes(StandardCharsets.UTF_8);
        byte [] saltedPass = new byte[passBytes.length+salt.length];
        System.arraycopy(passBytes, 0, saltedPass, 0, passBytes.length);
        System.arraycopy(salt, 0, saltedPass, passBytes.length, salt.length);
        //Hashear los bytes de la concatenacion: password || salt (en bytes)
        hashedbytes = sha256.digest(saltedPass);
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    if(hashedbytes!=null){
        //transformar el resultado de byte[] a String en hexadecimal
        hash = bytesToHex(hashedbytes);
    }
    return hash;
}
```

Figura 33: Código del método para hashear mediante SHA-256 con salt

Con estos ejemplos se ha podido observar como emplear las funciones hash ofrecidas por Java. Adicionalmente, se pueden emplear librerías de terceros como:

- Guava library
- Apache Comons Codecs
- Bouncy Castle Library

Una vez se han cubierto estas funciones hash se puede proveer de un mínimo de seguridad en la aplicación o sistema desarrollado. Cabe destacar que a la hora de almacenar la credencial de usuario es indispensable almacenar la clave y el *salt* empleado para obtener el *digest*, si no se almacena el *salt* en claro en la zona de almacenamiento del sistema, generalmente una base de datos MySQL, no se podrá realizar la comprobación para la autentificación.

Por último, se mostrará cómo emplear la nueva función hash ganadora del PHC (*Password Hashing Competition*) y considerada más segura criptográficamente Argon2 [55] en Java.

Actualmente, existen dos proyectos que permiten el uso de Argon2 en Java que son argon2-jvm [56] y jargon2-api [57]. Para este ejercicio se empleará el proyecto de argon2-jvm, se puede encontrar una plantilla que aporta las funciones básicas de este proyecto en OWASP [58].

Para hacer uso del proyecto es necesario incluir las dependencias correspondientes, en la guía del proyecto el autor propone el uso de las herramientas Maven o Gradle. En este caso se empleará Maven con las librerías de Argon2 precompiladas. En caso de querer usar la versión que no tiene las librerías precompiladas puede ser de ayuda la siguiente guía [59] aportada por el autor del proyecto.

La dependencia que se debe añadirse al pom.xml es la siguiente:

```
<dependency>
    <groupId>de.mkammerer</groupId>
    <artifactId>argon2-jvm</artifactId>
    <version>2.4</version>
</dependency>
```

Figura 34: Dependencias de Maven

Para emplear las clases desarrolladas en el proyecto, se debe importar el paquete *"de.mkammerer.argon2.*"*. Antes de pasar a la creación de los métodos para emplear Argon2, se explicará brevemente las clases del proyecto y las funcionalidades que ofrece.

El proyecto posee dos interfaces que son Argon2 y Argon2Advanced. La primera de las interfaces aporta las funciones de hash y verificación. El valor establecido por defecto para el *salt* es de 16 bytes, que es el tamaño mínimo recomendado por los autores de Argon2, y se genera a partir de la clase *SecureRandom* de Java. El tamaño de salida del hash es de 32 bytes, tanto el valor del *salt* como el de la longitud de salida se puede comprobar en la clase "*Argon2Constants.java"*. En el caso de la interfaz Argon2Advanced, se modifica el método que realiza el hash y éste permite introducir un *salt* propio.

Para la generación de los objetos Argon2 y Argon2Advanced, se emplea la clase "*Argon2Factory.java*". En esta clase se pueden obtener los objetos con los valores por defecto, se puede especificar dentro de Argon2 y Argon2Advanced el tipo de algoritmo, el tipo de algoritmo se refiere a *Argon2i, Argon2d y Argon2id*, y por último se puede modificar el tamaño

de salida de la función y la longitud del hash a emplear. A continuación, se muestran unos ejemplos de los métodos para generar las distintas instancias de Argon2.

```
/*Crear instancia de Argon2 por defecto, que es Argon2i*/
private static Argon2 createInstanceArgon2(){
    return Argon2Factory.create();
}
/*Crear instancia de Argon2i
 * Sirve preferentemente para hashear contraseñas y derivar claves, segun los autores*/
private static Argon2 createInstanceArgon2i(){
    return Argon2Factory.create(Argon2Factory.Argon2Types.ARGON2i);
}

/*Crear instancia de Argon2d
 * Sirve preferentemente para la generación de criptomonedas
 * y en aplicaciones que no se esperen ataques de canal lateral*/
private static Argon2 createInstanceArgon2d(){
    return Argon2Factory.create(Argon2Factory.Argon2Types.ARGON2d);
}

/*Crear instancia de Argon2id
 * Combinación de los dos modos, actualmente recomendado por el borrador de la RFC de Argon2
 * para servicios de autentificación y opción por defecto*/
private static Argon2 createInstanceArgon2id(){
    return Argon2Factory.create(Argon2Factory.Argon2Types.ARGON2id);
}

//Crear instancia de Argon2i personalizada
private static Argon2 createInstanceArgon2i(int saltlength, int outputlength){
    return Argon2Factory.create(Argon2Factory.Argon2Types.ARGON2i,saltlength, outputlength);
}

//Crear instancia de Argon2i con la interfaz Argon2Advanced
private static Argon2Advanced createAdvancedInstanceArgon2i(){
    return Argon2Factory.createAdvanced(Argon2Factory.Argon2Types.ARGON2i);
}
```

Figura 35: Codigo de métodos para instanciar objetos de Argon2

En esta captura se pueden apreciar las distintas formas de instanciar los objetos. En los casos de *Argon2d y Argon2id*, también se pueden emplear con la versión personalizada y con Argon2Advanced.

El siguiente paso es realizar los métodos *protect()* para obtener el hash la contraseña. Para ello se debe especificar los parámetros que va a emplear el algoritmo Argon2 y el tipo que se usará. Se mostrarán varias posibles implementaciones del método *protect()* dependiendo del método de hash escogido o si se emplea la clase auxiliar Argon2Helper. El primer ejemplo es el método *protectBasic()* en el que se especifican los parámetros que necesita Argon2.

```java
/*Método protect especificando los parámetros a emplear
 *  password: String que se hashea
 *  iterations: iteraciones del algoritmo
 *  memory: memoria a emplear
 *  thread: grado de paralelización
 *  type: tipo de Argon2, 0->Argon2i ; 1->Argon2d ; 2->Argon2id*/
public static String protectBasic(String password, int iterations, int memory, int threads, int type){
    Argon2 argon2hasher= null;
    String argon2hash;
    try{
        //Crear instancia de Argon2
        if(type == 0){
            argon2hasher = createInstanceArgon2i();
        }else if(type == 1){
            argon2hasher = createInstanceArgon2d();
        }else if(type == 2){
            argon2hasher = createInstanceArgon2id();
        }else{
            System.out.println("Unkown option, setted to Argon2i due to authors reccomendation");
            argon2hasher = createInstanceArgon2i();
        }
        /*Hashear con los valores por defecto de Argon2
         * Salt de 16bytes
         * Longitud del hash de salida 32 bytes
         * */
        argon2hash = argon2hasher.hash(iterations, memory, threads, password);
    }finally{
        //Nada que hacer
    }
    return argon2hash;
}
```

Figura 36: Código del método protectBasic

El siguiente ejemplo muestra el método *protect()* en el que se hace uso de la clase Argon2Helper para ayudar a escoger el número de iteraciones que debe emplear Argon2. En este caso se deben especificar el número máximo de milisegundos que se quiere que Argon2 tarde en obtener el hash.

```java
/*Método protect que calcula las iteraciones dependiendo de los milisegundos especificados
 *  password: String que se hashea
 *  miliseconds: máximo número de milisegundos que tardará el algoritmo
 *  memory: memoria a emplear
 *  thread: grado de paralelización
 *  type: tipo de Argon2, 0->Argon2i ; 1->Argon2d ; 2->Argon2id*/
public static String protect(String password, int miliseconds, int memory, int threads, int type){
    Argon2 argon2hasher= null;
    String argon2hash;
    int iterations;
    try{
        //Crear instancia de Argon2
        if(type == 0){
            argon2hasher = createInstanceArgon2i();
        }else if(type == 1){
            argon2hasher = createInstanceArgon2d();
        }else if(type == 2){
            argon2hasher = createInstanceArgon2id();
        }else{
            System.out.println("Unkown option, setted to Argon2i due to authors reccomendation");
            argon2hasher = createInstanceArgon2i();
        }
        //Obtener número de iteraciones óptimo para el sistema teniendo en cuenta el tiempo máximo
        iterations = Argon2Helper.findIterations(argon2hasher, miliseconds, memory, threads);
        /*Hashear con los valores por defecto de Argon2
         * Salt de 16 bytes
         * Longitud del hash de salida 32 bytes*/
        argon2hash = argon2hasher.hash(iterations, memory, threads, password);
    }finally{
        //Nada que hacer
    }
    return argon2hash;
}
```

Figura 37: Código del método protect

Hasta ahora en los métodos se han empleado las instancias de Argon2 con los valores por defecto de longitud del hash y longitud del *salt.* En este ejemplo, se puede especificar la longitud de salida y del *salt*. Adicionalmente, se emplea la clase Argon2Helper. Este método puede modificarse para no emplear la clase Argon2Helper y se especifique las iteraciones por parámetro.

```
//Metodo protect() que permite modificar la longitud del salt y de salida de la función
public static String protectTuned(String password, int miliseconds, int memory, int threads, int type, int outputlength, int saltlength){
    Argon2 argon2hasher= null;
    String argon2hash;
    int iterations;
    try{
        //Crear instancia de Argon2
        if(type == 0){
            argon2hasher = createInstanceArgon2i(saltlength,outputlength);
        }else if(type == 1){
            argon2hasher = createInstanceArgon2d(saltlength,outputlength);
        }else if(type == 2){
            argon2hasher = createInstanceArgon2id(saltlength,outputlength);
        }else{
            System.out.println("Unkown option, setted to Argon2i due to authors reccomendation");
            argon2hasher = createInstanceArgon2i(saltlength,outputlength);
        }
        //Obtener número de iteraciones óptimo para el sistema teniendo en cuenta el tiempo máximo
        iterations = Argon2Helper.findIterations(argon2hasher, miliseconds, memory, threads);
        //Hashear la contraseña
        argon2hash = argon2hasher.hash(iterations, memory, threads, password);
    }finally{
        //Nada que hacer
    }
    return argon2hash;
}
```

Figura 38: Código del método protectTuned

Todos estos métodos han recibido la contraseña como *String*, pero es posible emplear un método que reciba la contraseña como cadena de caracteres y cuando se genere el hash borrarla de memoria, para así evitar posibles ataques que lean de memoria el valor de la contraseña porque el recolector de basura no haya decidido eliminar esa variable. Este método sería más seguro debido al borrado de la contraseña de memoria.

```
//Metodo igual que protectBasic(), pero más seguro porque se wipea la contraseña de memoria
public static String protectBasic(char[] password, int iterations, int memory, int threads, int type){
    Argon2 argon2hasher= null;
    String argon2hash;
    try{
        //Crear instancia de Argon2
        if(type == 0){
            argon2hasher = createInstanceArgon2i();
        }else if(type == 1){
            argon2hasher = createInstanceArgon2d();
        }else if(type == 2){
            argon2hasher = createInstanceArgon2id();
        }else{
            System.out.println("Unkown option, setted to Argon2i due to authors reccomendation");
            argon2hasher = createInstanceArgon2i();
        }
        /*Hashear con los valores por defecto de Argon2
         * Salt de 16bytes
         * Longitud del hash de salida 32 bytes
         * */
        argon2hash = argon2hasher.hash(iterations, memory, threads, password);
    }finally{
        if(argon2hasher != null){
            //Eliminar la contraseña de memoria
            argon2hasher.wipeArray(password);
        }
    }
    return argon2hash;
}
```

Figura 39: Código del método protectBasic con borrado de la contraseña de memoria

Tal y como se ha mostrado en los ejemplos anteriores, se puede modificar este método para que emplee la clase Argon2Helper o emplee los constructores de las instancias con parámetros para la longitud de hash y *salt* deseados.

El último de los ejemplos de la implementación del método *protect()* hará uso del objeto Argon2Advanced. En este caso el resultado devuelto es el array de bytes del hash generado. Este método permite la inclusión de un *salt* propio.

```
/*Método protect especificando los parámetros a emplear
 *  password: String que se hashea
 *  iterations: iteraciones del algoritmo
 *  memory: memoria a emplear
 *  thread: grado de paralelización
 *  salt: bytes de salt que se emplearan en el proceso
 *  type: tipo de Argon2, 0->Argon2i ; 1->Argon2d ; 2->Argon2id*/
public static byte[] protectBasicSalted(String password, int iterations, int memory, int threads, byte[] salt, int type){
    Argon2Advanced argon2hasher= null;
    byte[] argon2hash;
    try{
        //Crear instancia de Argon2
        if(type == 0){
            argon2hasher = createAdvancedInstanceArgon2i();
        }else if(type == 1){
            argon2hasher = createAdvancedInstanceArgon2d();
        }else if(type == 2){
            argon2hasher = createAdvancedInstanceArgon2id();
        }else{
            System.out.println("Unkown option, setted to Argon2i due to authors recommendation");
            argon2hasher = createAdvancedInstanceArgon2i();
        }
        /*Hashear con los valores por defecto de Argon2
         * Longitud del hash de salida 32 bytes*/
        argon2hash = argon2hasher.rawHash(iterations, memory, threads, password, salt);
    }finally{
        //Nada que hacer
    }
    return argon2hash;
}
```

Figura 40: Código del método protectBasicSalted

Al igual que en ejemplos anteriores, se puede modificar este método para que use instancias del objeto Argon2Advanced sin los valores predeterminados, también se puede modificar el tipo de dato de la contraseña recibida para luego borrarla de memoria.

Una vez se ha mostrado las distintas posibles implementaciones del método *protect()* se procede a desarrollar el método *verify()* que servirá para comprobar si la clave es correcta. Previa a la demostración del método *verify()* se explicará el formato de salida de Argon2. Para esta explicación se empleará un resultado obtenido con los métodos propuestos.

```
$argon2i$v=19$m=65536,t=25,p=2$CWHoOl4AM/M/9qmHS0kbTg$Du8aJ5PsAa2OuGP16shAUg
```

La estructura genérica del output es:

$argon2{*type*}$v={*version*}$m={*value*},t={ *value* },p={ *value* }${*salt*}${*hash*}

El separador empleado para discernir entre distintas secciones es el símbolo $. El primer campo es argon2{type}, en el que *type* puede ser i, d o id haciendo referencia a los tipos de Argon2 que son Argon2i, Argon2d, Argon2id. El siguiente campo es v={*version*}, donde *version* es la versión de Argon2 que actualmente es 0x13, en decimal es 19. El siguiente campo contiene los valores de los parámetros empleados para la función donde m, t y p hacen referencia a la memoria, iteraciones y grado de paralelización empleados. El campo {*value*} es el valor empleado para el parámetro correspondiente. El campo siguiente, {*salt*} es la representación en base64 del *salt*. El último campo, {*hash*}, es la representación en base64 del hash. El siguiente ejemplo muestra una implementación del método *verify()*.

```java
//Método para verificar el hash con la contraseña introducida, se necesita saber el tipo de Argon2 empleado
public static boolean verify(String hash, String password, int type){
    Argon2 argon2hasher = null;
    boolean matches;
    try{
        //Crear instancia de Argon2
        if(type == 0){
            argon2hasher = createInstanceArgon2i();
        }else if(type == 1){
            argon2hasher = createInstanceArgon2d();
        }else if(type == 2){
            argon2hasher = createInstanceArgon2id();
        }else{
            System.out.println("Unkown option, setted to Argon2i due to authors reccomendation");
            argon2hasher = createInstanceArgon2i();
        }
        //Verificar el hash
        matches = argon2hasher.verify(hash, password);
    }finally{
        //Nada que hacer
    }
    return matches;
}
```

Figura 41: Código del método verify

Este método puede cambiarse el argumento de *password* para que sea de tipo array de caracteres, en caso de hacerlo, se debe añadir el borrado de memoria de la contraseña en el bloque *finally*.

La siguiente implementación es empleando el método *verify()* cuando se emplea Argon2 con *salt* y tamaño de salida de la función hash.

```java
public static boolean verifyTuned(String hash, String password, int type, int saltlength, int outputlength){
    Argon2 argon2hasher = null;
    boolean matches;
    try{
        //Crear instancia de Argon2
        if(type == 0){
            argon2hasher = createInstanceArgon2i(saltlength,outputlength);
        }else if(type == 1){
            argon2hasher = createInstanceArgon2d();
        }else if(type == 2){
            argon2hasher = createInstanceArgon2id();
        }else{
            System.out.println("Unkown option, setted to Argon2i due to authors recommendation");
            argon2hasher = createInstanceArgon2i();
        }
        //Verificar el hash
        matches = argon2hasher.verify(hash, password);
    }finally{
        //Nada que hacer
    }
    return matches;
}
```

Figura 42: Código del método verifyTuned

Al igual que en el método anterior, puede modificarse para que el parámetro *password* sea un vector de caracteres.

Como trabajo final, se expone la creación de la clase *PasswordTreatment.java* en la que se implementan los métodos de *protect()* y *verify()* empleando Argon2 de la manera más personalizable posible. Se propone la creación de los métodos para poder emplear cualquiera de los tres tipos distintos de Argon2.

```java
//Crear instancia de Argon2i personalizada
private static Argon2 createInstanceArgon2i(int saltlength, int outputlength){
    return Argon2Factory.create(Argon2Factory.Argon2Types.ARGON2i,saltlength, outputlength);
}

//Crear instancia de Argon2d personalizada
private static Argon2 createInstanceArgon2d(int saltlength, int outputlength){
    return Argon2Factory.create(Argon2Factory.Argon2Types.ARGON2d,saltlength, outputlength);
}

//Crear instancia de Argon2id personalizada
private static Argon2 createInstanceArgon2id(int saltlength, int outputlength){
    return Argon2Factory.create(Argon2Factory.Argon2Types.ARGON2id,saltlength, outputlength);
}
```

Figura 43: Código de los métodos para instanciar Argon2 (PasswordTreatment)

También se proporcionan dos implementaciones distintas del método *protect()*, en una se especifican todos los parámetros que necesita Argon2 de originalmente, y en la otra se emplea la clase auxiliar Argon2Helper para que realice el cálculo de las iteraciones para el sistema. Ambos métodos recibirán el parámetro de la contraseña como cadena de caracteres para poder eliminar de la memoria el valor una vez se haga la función hash.

```java
/*Método protect especificando los parámetros a emplear
 *  password: String que se hashea
 *  iterations: iteraciones del algoritmo
 *  memory: memoria a emplear
 *  thread: grado de paralelización
 *  type: tipo de Argon2, 0->Argon2i ; 1->Argon2d ; 2->Argon2id
 *  saltlength: longitud del salt a emplear
 *  outputlength: longitud del hash de salida*/
public static String protect(char[] password, int iterations, int memory, int threads, int type, int saltlength, int outputlength){
    Argon2 argon2hasher= null;
    String argon2hash;
    try{
        //Crear instancia de Argon2
        switch(type){
            case 0:
                argon2hasher = createInstanceArgon2i(saltlength,outputlength);break;
            case 1:
                argon2hasher = createInstanceArgon2d(saltlength,outputlength);break;
            case 2:
                argon2hasher = createInstanceArgon2id(saltlength,outputlength);break;
            default:
                System.out.println("Unkown option, setted to Argon2i due to authors recommendation");
                argon2hasher = createInstanceArgon2i(saltlength,outputlength);
        }
        //Hashear la contraseña
        argon2hash = argon2hasher.hash(iterations, memory, threads, password);
    }finally{
        //Borrar contraseña de memoria
        if(argon2hasher != null)
            argon2hasher.wipeArray(password);
    }
    return argon2hash;
}
```

Figura 44: Código del método protect (PasswordTreatment)

```java
//Método similar a protect(), solo que calcula las iteraciones para que el algoritmo de Argon2
//tarde como máximo 'miliseconds' milisegundos en calcular el hash
public static String protectAided(char[] password, int miliseconds, int memory, int threads, int type, int saltlength, int outputlength){
    Argon2 argon2hasher= null;
    String argon2hash;
    int iterations;
    try{
        //Crear instancia de Argon2
        switch(type){
            case 0:
                argon2hasher = createInstanceArgon2i(saltlength,outputlength);break;
            case 1:
                argon2hasher = createInstanceArgon2d(saltlength,outputlength);break;
            case 2:
                argon2hasher = createInstanceArgon2id(saltlength,outputlength);break;
            default:
                System.out.println("Unkown option, setted to Argon2i due to authors recommendation");
                argon2hasher = createInstanceArgon2i(saltlength,outputlength);
        }
        //Obtener número de iteraciones óptimo para el sistema teniendo en cuenta el tiempo máximo
        iterations = Argon2Helper.findIterations(argon2hasher, miliseconds, memory, threads);
        //Hashear la contraseña
        argon2hash = argon2hasher.hash(iterations, memory, threads, password);
    }finally{
        //Borrar contraseña de memoria
        if(argon2hasher != null)
            argon2hasher.wipeArray(password);
    }
    return argon2hash;
}
```

Figura 45: Código del método protectAided (PasswordTreatment)

Después de los métodos *protect(),* se procede a la implementación del método *verify()* para poder verificar las contraseñas.

```java
//Método para verificar el hash con la contraseña introducida, se necesita saber el tipo de Argon2 empleado
public static boolean verify(String hash, char[] password, int type, int saltlength, int outputlength){
    Argon2 argon2hasher = null;
    boolean matches;
    try{
        //Crear instancia de Argon2
        switch(type){
            case 0:
                argon2hasher = createInstanceArgon2i(saltlength,outputlength);break;
            case 1:
                argon2hasher = createInstanceArgon2d(saltlength,outputlength);break;
            case 2:
                argon2hasher = createInstanceArgon2id(saltlength,outputlength);break;
            default:
                System.out.println("Unkown option, setted to Argon2i due to authors recommendation");
                argon2hasher = createInstanceArgon2i(saltlength,outputlength);
        }
        //Verificar el hash
        matches = argon2hasher.verify(hash, password);
    }finally{
        //Borrar contraseña de memoria
        if(argon2hasher != null)
            argon2hasher.wipeArray(password);
    }
    return matches;
}
```

Figura 46: Código del método verify (PasswordTreatment)

Por último, mostrar una ejecución simple en el *main* para comprobar que los métodos funcionan.

```java
    public static void main(String[] args) {
        String pass = "password123";
        //Argon2i, con un máximo de 1000 milisegundos. Una longitud de 16 bytes de salt y 16 bytes de longitud de salida
        String argoned = protectAided(pass.toCharArray(),1000,65536,2,0,16,16);
        System.out.println(argoned);
        System.out.println("Verification is: "+verify(argoned,pass.toCharArray(),0,16,16));
    }

}
```

```
Problems  Javadoc  Declaration  Console
<terminated> PasswordTreatment [Java Application] C:\Program Files\Java\jre1.8.0_162\bin\javaw.exe (14 sept. 2018 14:43:52)
$argon2i$v=19$m=65536,t=25,p=2$qhVVoUmtvjrgw8VGVpJeMg$TkHY4hFhAm8aA8VE2HD6pw
Verification is: true
```

Figura 47: Verificación sencilla de un valor hash (PasswordTreatment)

Esta demostración del uso de Argon2 está orientada a Java, pero tal y como se mencionó al inicio se puede emplear para otros lenguajes. Sólo hay que comprobar el listado que aparece en el repositorio original [55] en la sección *Bindings*.

Finalmente, hay que mencionar que en caso de no saber qué tipo de Argon2 se debe emplear, se puede comprobar el borrador de la RFC [60]. Adicionalmente, para saber cuáles son los parámetros recomendados para Argon2 se puede comprobar el apartado 9 del documento de especificaciones [61].

4.2.4. Material Didáctico 4. Cifrado simétrico, asimétrico e híbrido.

Este ejercicio se centra en discutir las propiedades de cada uno de los tipos de cifrado existentes, sus ventajas, sus inconvenientes y en qué casos se debe usar cada uno de los distintos métodos de cifrado.

En caso de haber escogido el cifrado de los datos como solución para proteger las credenciales de usuario, es necesario saber qué diferencias existen entre el cifrado simétrico y asimétrico. El cifrado simétrico se basa en el uso de una única clave secreta que es conocida por las entidades responsables del cifrado y descifrado, permanece desconocida para el resto del mundo. El cifrado simétrico emplea la misma clave para cifrar y descifrar, por ello es necesaria la protección de la clave. La principal ventaja del cifrado simétrico es la rapidez respecto al cifrado asimétrico, así como el uso de menos recursos del sistema.

El otro tipo de cifrado es el cifrado asimétrico también conocido como cifrado de clave pública. Este cifrado se basa en el uso de dos claves, una de ellas se llama clave pública y como indica su nombre y es conocida por todo el mundo. La otra clave se llama clave privada y esta debe ser secreta y sólo ser conocida por el propietario del par de claves. El proceso de cifrado del sistema asimétrico se centra en el cifrado de los datos empleando la clave pública del destinatario y el descifrado de los mismos empleando la clave privada correspondiente. Las principales ventajas del cifrado asimétrico es que es bastante difícil de romper y bastante potente ya que se requieren las dos claves para el cifrado y descifrado. Los inconvenientes respecto al cifrado simétrico es que es bastante más lento y emplea muchos más recursos.

Existe otro tipo de cifrado que deriva del uso de los dos cifrados mencionados previamente. Este cifrado se llama cifrado híbrido y se encarga de emplear las ventajas del cifrado simétrico para poder cifrar los datos de forma rápida y sin gastar demasiados recursos del sistema mientras que gracias al sistema de cifrado asimétrico se mantiene a salvo la clave secreta empleada en el cifrado simétrico. De esta forma se cifran los datos con gran eficiencia y la clave simétrica está protegida por el cifrado asimétrico que es un sistema bastante robusto.

Teniendo estas cualidades en cuenta, hay que recordar que existen distintos modos de operación con los sistemas de cifrado por bloques, entre los que se encuentran los recomendados por el NIST [15] que son:

- ECB, *Electronic Code Book Mode*
- CBC, *Cipher Block Chaining Mode*
- CFB, *Cipher Feedback Mode*
- OFB, *Output Feedback Mode*
- CTR, *Counter Mode*

Una explicación más detallada de cada modo de operación se encuentra en el apartado 2.2 Propuesta de soluciones.

Para la realización de este ejercicio se llevará a cabo una implementación del cifrado simétrico y asimétrico en Android Studio, ya que se ha observado que es una de las opciones más elegidas para desarrollar aplicaciones móviles en los últimos años. Se hará uso de la herramienta AVD (*Android Virtual Device*) para la simulación de un móvil Nexus 5 con Android 8.0 Oreo. Se empleará un proyecto con una única actividad provista de un *EditText* y un *Button* para invocar a los métodos de cifrado empleando el texto introducido en *EditText* cuando se haga click en el botón.

Primero se muestran los métodos empleados para el cifrado simétrico, en este caso se ha escogido emplear el algoritmo AES, se puede comprobar qué algoritmos de cifrado y modos de cifrado están disponibles en la página de Android.developer [62], así como para la generación de claves secretas [63].

```
/*Obtener cifrador de AES*/
public Cipher getCipherAES() throws NoSuchPaddingException, NoSuchAlgorithmException {
    return Cipher.getInstance("AES_256/ECB/PKCS5Padding");
}

/*Generar clave secreta AES-256*/
public SecretKey generarClaveSecreta() throws NoSuchAlgorithmException {
    KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
    keyGenerator.init( keysize: 256);
    SecretKey secretKey = keyGenerator.generateKey();
    return secretKey;
}
```

Figura 48: Obtención del cifrador y clave para cifrador simétrico AES-256

El siguiente paso es la creación de los métodos para el cifrado y descifrado correspondiente.

```
/*Cifrado simetrico AES-256*/
public String cifradoSimetrico(SecretKey secretKey, String textoPlano){
    Log.d( tag: "Cifrado", msg: "cifrado SIMETRICO");
    Cipher cifrador;
    String res = null;
    try {
        cifrador = getCipherAES();
        cifrador.init(Cipher.ENCRYPT_MODE, secretKey);
        res = Base64.encodeToString(cifrador.doFinal(textoPlano.getBytes()), Base64.NO_WRAP);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return res;
}

/*Descifrado simetrico AES-256*/
public String descifradoSimetrico(SecretKey secretKey, String textoCifrado){
    Log.d( tag: "Cifrado", msg: "descifrado SIMETRICO");
    Cipher cifrador;
    try {
        cifrador = getCipherAES();
        cifrador.init(Cipher.DECRYPT_MODE, secretKey);
        return new String (cifrador.doFinal(Base64.decode(textoCifrado, Base64.NO_WRAP)));
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
```

Figura 49: Métodos de cifrado y descifrado de cifrador simétrico

Por último, se muestra el código empleado para el cifrado y descifrado de un String para comprobar que el resultado sigue siendo el mismo, para comprobar los resultados se emplearán los *Log*.

```
/*Cifrado SIMETRICO*/
SecretKey secretKey = null;
try {
    secretKey = generarClaveSecreta();
} catch (NoSuchAlgorithmException e) {
    e.printStackTrace();
}
texto_cifrado = cifradoSimetrico(secretKey,texto);
Log.d( tag: "Cifrado",  msg: "Texto cifrado SIMETRICO: "+texto_cifrado);
texto_descifrado = descifradoSimetrico(secretKey,texto_cifrado);
Log.d( tag: "Cifrado",  msg: "Texto descifrado SIMETRICO: "+texto_descifrado);
```

Figura 50: Código para loggear el resultado de cifrar y descifrar un string (Cifrador simétrico)

```
09-19 15:13:09.023 12964-12964/uc3m.tfg.com.tfg_credenciales_usuario D/Cifrado: Texto Descifrado ASIMETRICO: textoplano
09-19 15:13:09.025 12964-12964/uc3m.tfg.com.tfg_credenciales_usuario D/Cifrado: cifrado SIMETRICO
09-19 15:13:09.026 12964-12964/uc3m.tfg.com.tfg_credenciales_usuario D/Cifrado: Texto cifrado SIMETRICO: qOoTk0tTv9AltDf3ggEs0Q==
    descifrado SIMETRICO
09-19 15:13:09.027 12964-12964/uc3m.tfg.com.tfg_credenciales_usuario D/Cifrado: Texto descifrado SIMETRICO: textoplano
```

Figura 51: Resultado de cifrar y descifrar un string (Cifrador simétrico)

El siguiente ejemplo se utilizará el cifrado asimétrico con RSA, para ello se empleará *AndroidKeyStore* [64] para que se encargue del almacenamiento seguro de las claves. Primero se muestran los métodos que cargan las claves del almacén de claves y la generación del par de claves.

```java
/*Metodos para obtener el par de claves de AndroidKeyStore*/
public void loadKeyStore() {
    try {
        keyStore = KeyStore.getInstance(ANDROID_KEYSTORE);
        keyStore.load( param: null);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

/*Metodo para generar el par de claves publica-privada*/
public void generateNewKeyPair(String alias, Context context) throws Exception {
    Calendar start = Calendar.getInstance();
    Calendar end = Calendar.getInstance();
    // expires 1 year from today
    end.add(Calendar.YEAR, i1: 1);
    KeyPairGeneratorSpec spec = new KeyPairGeneratorSpec.Builder(context)
            .setAlias(alias)
            .setSubject(new X500Principal( name: "CN=" + alias))
            .setSerialNumber(BigInteger.TEN)
            .setStartDate(start.getTime())
            .setEndDate(end.getTime())
            .build();
    // use the Android keystore
    KeyPairGenerator gen = KeyPairGenerator.getInstance( algorithm: "RSA",ANDROID_KEYSTORE);
    gen.initialize(spec);
    // generates the keypair
    gen.generateKeyPair();
}
```

Figura 52: Métodos para obtener par de claves asimétricas

Los siguientes métodos son la implementación de la obtención de las claves públicas y privadas del *KeyStore* recuperado previamente. Los métodos para cargar la clave pública y privada se emplearán en el cifrado y descifrado respectivamente.

```java
/*Obtener clave publica*/
public PublicKey loadPublicKey(String alias) throws Exception {
    KeyStore.Entry entry = keyStore.getEntry(alias, protParam: null);
    if (!(entry instanceof KeyStore.PrivateKeyEntry)) {
        Log.d( tag: "KeyStore", msg: " alias: " + alias + " is not a PrivateKey");
        return null;
    }
    PublicKey publicKey = (PublicKey) ((KeyStore.PrivateKeyEntry)entry).getCertificate().getPublicKey();
    return publicKey;
}

/*Obtener clave privada*/
public PrivateKey loadPrivateKey(String alias) throws Exception {
    KeyStore.Entry entry = keyStore.getEntry(alias, protParam: null);
    if (!(entry instanceof KeyStore.PrivateKeyEntry)) {
        Log.d( tag: "KeyStore", msg: " alias: " + alias + " is not a PrivateKey");
        return null;
    }
    PrivateKey privateKey = (PrivateKey) ((KeyStore.PrivateKeyEntry)entry).getPrivateKey();
    return privateKey;
}
```

Figura 53: Métodos para recuperar las claves públicas o privadas

La implementación de los métodos de cifrado y descifrado con RSA son los siguientes, así como la obtención del cifrador para el algoritmo RSA.

```java
/*Obtener cifrador de RSA*/
public Cipher getCipherRSA() throws NoSuchPaddingException, NoSuchAlgorithmException {
    return Cipher.getInstance("RSA/NONE/PKCS1Padding");
}
```

Figura 54: Método para obtener el cifrador asimétrico

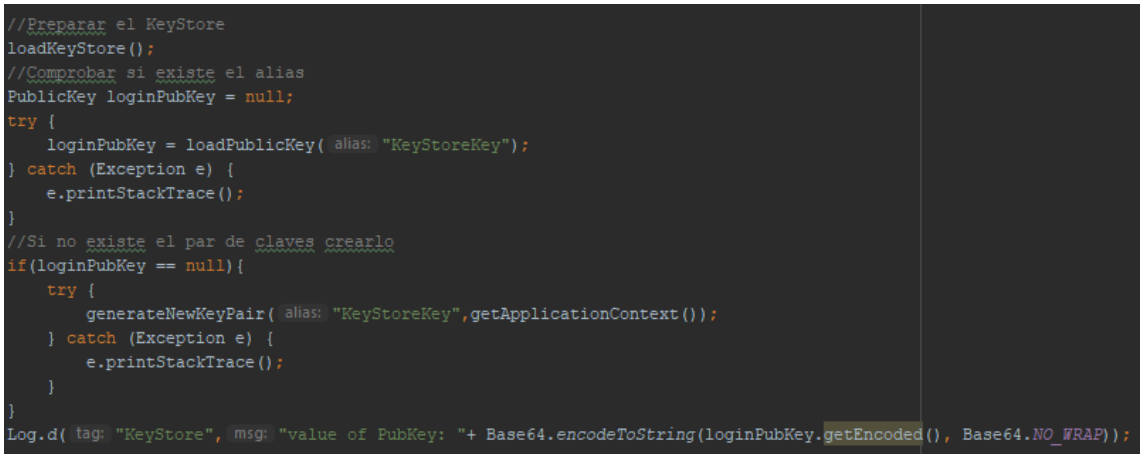
```java
/*Cifrado con RSA*/
public String cifradoRSA(String alias, String texto){
    Log.d( tag: "Cifrado", msg: "cifradoRSA");
    Cipher cifrador;
    String res = null;
    try {
        cifrador = getCipherRSA();
        //Obtener la clave publica para cifrar
        PublicKey publicKey = loadPublicKey(alias);
        cifrador.init(Cipher.ENCRYPT_MODE, publicKey);
        res = Base64.encodeToString(cifrador.doFinal(texto.getBytes()), Base64.NO_WRAP);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return res;
}
```

Figura 55: Método para cifrar con el cifrador asimétrico

```java
/*Descifrado con RSA*/
public String descifradoRSA(String alias, String textoCifrado){
    Log.d( tag: "Cifrado", msg: "descifradoRSA");
    Cipher cifrador;
    try {
        cifrador = getCipherRSA();
        //Obtener la clave privada para descifrar
        PrivateKey privateKey = loadPrivateKey(alias);
        if(privateKey == null){
            Log.d( tag: "KeyStore", msg: "no hay privateKey");
        }
        cifrador.init(Cipher.DECRYPT_MODE, privateKey);
        return new String (cifrador.doFinal(Base64.decode(textoCifrado, Base64.NO_WRAP)));
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
```

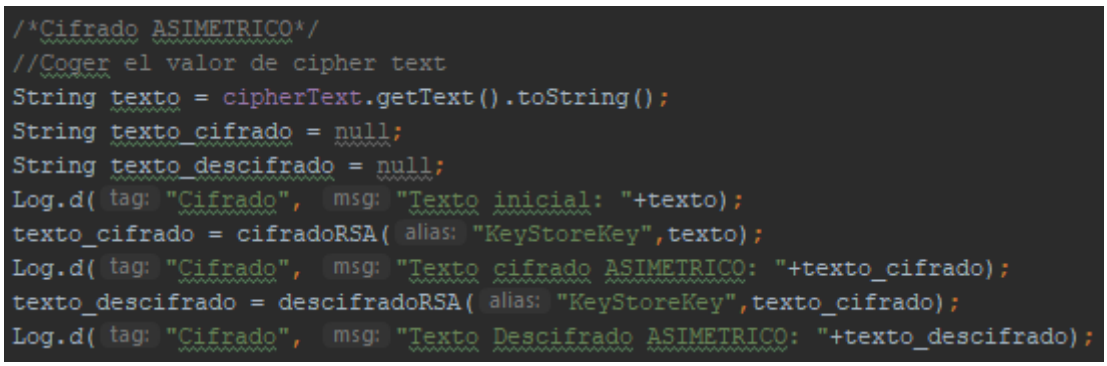
Figura 56: Método para descifrar con el cifrador asimétrico

Por último, falta por mostrar el código empleado para cargar el KeyStore en la actividad, así como el resultado del cifrado y descifrado del texto en plano.



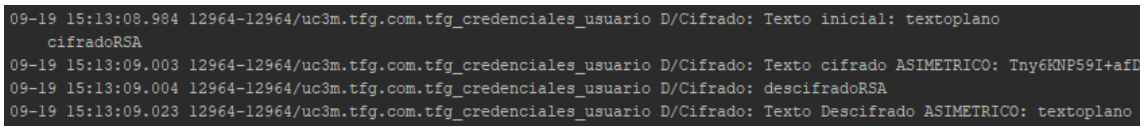
```
//Preparar el KeyStore
loadKeyStore();
//Comprobar si existe el alias
PublicKey loginPubKey = null;
try {
    loginPubKey = loadPublicKey( alias: "KeyStoreKey");
} catch (Exception e) {
    e.printStackTrace();
}
//Si no existe el par de claves crearlo
if(loginPubKey == null){
    try {
        generateNewKeyPair( alias: "KeyStoreKey",getApplicationContext());
    } catch (Exception e) {
        e.printStackTrace();
    }
}
Log.d( tag: "KeyStore", msg: "value of PubKey: "+ Base64.encodeToString(loginPubKey.getEncoded(), Base64.NO_WRAP));
```

Figura 57: Código para generar el par de claves



```
/*Cifrado ASIMETRICO*/
//Coger el valor de cipher text
String texto = cipherText.getText().toString();
String texto_cifrado = null;
String texto_descifrado = null;
Log.d( tag: "Cifrado",  msg: "Texto inicial: "+texto);
texto_cifrado = cifradoRSA( alias: "KeyStoreKey",texto);
Log.d( tag: "Cifrado",  msg: "Texto cifrado ASIMETRICO: "+texto_cifrado);
texto_descifrado = descifradoRSA( alias: "KeyStoreKey",texto_cifrado);
Log.d( tag: "Cifrado",  msg: "Texto Descifrado ASIMETRICO: "+texto_descifrado);
```

Figura 58: Código para cifrar y descifrar con cifrador asimétrico



```
09-19 15:13:08.984 12964-12964/uc3m.tfg.com.tfg_credenciales_usuario D/Cifrado: Texto inicial: textoplano
    cifradoRSA
09-19 15:13:09.003 12964-12964/uc3m.tfg.com.tfg_credenciales_usuario D/Cifrado: Texto cifrado ASIMETRICO: Tny6KNP59I+afD
09-19 15:13:09.004 12964-12964/uc3m.tfg.com.tfg_credenciales_usuario D/Cifrado: descifradoRSA
09-19 15:13:09.023 12964-12964/uc3m.tfg.com.tfg_credenciales_usuario D/Cifrado: Texto Descifrado ASIMETRICO: textoplano
```

Figura 59: Resultado de cifrar y descifrar con cifrador asimétrico

En esta última captura, la longitud del valor del texto cifrado es mayor del que se podía mostrar, es decir, que el valor mostrado en la captura no es el valor completo del cifrado. Aun así, se puede comprobar que el descifrado del texto da como lugar al texto original.

Con estos ejemplos termina la presentación de la implementación del cifrado simétrico y asimétrico en Android Studio en clases .java. Para aplicar el cifrado híbrido mencionado al inicio del ejercicio, se podría generar la clave secreta simétrica y almacenarla cifrada con el sistema asimétrico en un *SharedPreferences*, de esta forma, aunque se pueda leer la clave secreta estaría cifrada con el sistema asimétrico. En caso de necesitar cifrar algún dato se recuperaría la clave simétrica y se descifraría, luego se emplearían los métodos de cifrado y descifrado simétrico según convenga.

4.2.5. Material Didáctico 5. Whitelist & Blacklist mediante expresiones regulares.

En este ejercicio se explicarán los conceptos de las expresiones regulares y como se pueden aplicar en Java para crear métodos que sirvan como *whitelists* y *blacklists* aportando así cierto nivel de protección para evitar ataques como la inyección SQL, *Cross Site Scripting* y el uso de caracteres no permitidos en la aplicación. El uso de estas *whitelists* y *blacklists* tiene el objetivo principal de cubrir la comprobación de los campos de los formularios de registro y la comprobación a la hora de generar unas nuevas credenciales de usuario en el sistema.

El término *whitelist* se refiere a todo lo que se permite o se considera una opción válida. En este caso se centrará el concepto de *whitelist* en los caracteres aceptables que tendrá el sistema. El término *blacklist* se refiere a aquello que está prohibido o no se considera válido. Las *blacklist* se orientarán a los caracteres que no están permitidos para prevenir posibles ataques de inyección SQL y *Cross Site Scripting*. Simplificando la explicación de los conceptos, se puede decir que las *whitelists* son todo lo que se permite y las *blacklist* son todo lo que no se permite.

Este ejercicio se realizará en Java y se emplearán principalmente las clases *Pattern* y *Matcher* para el diseño de los métodos mediante expresiones regulares también llamadas *regex*. La clase *Pattern* se emplea para compilar las expresiones regulares en un objeto de tipo *Pattern*, mediante *Pattern.compile().* Además de especificar la expresión regular que se va a compilar, se pueden añadir parámetros opcionales como puede ser *Pattern.CASE_INSENSITIVE* para que no se distinga entre mayúsculas o minúsculas al comprobar la expresión regular contra el input. La otra clase que se va a emplear es la clase *Matcher* que es la que realmente se encarga de realizar la comprobación del input contra el patrón especificado por el objeto de tipo *Pattern*. El objeto *Matcher* se instancia a partir del objeto *Pattern*. Generalmente, se empleará el método *matches()* para comprobar que la expresión regular se cumple, pero existen otros métodos que permiten contar y reemplazar partes del input que se correspondan con el patrón.

A continuación, se explicará brevemente los elementos que se emplean para generar las expresiones regulares. Esta tabla contiene los caracteres que se emplearan, si se desea conocer todos los caracteres que proporciona Java se puede comprobar aquí [65].

Limitadores y cuantificadores

Definen el inicio y final de las expresiones regulares, así como el número de veces que debe aparecer un carácter o expresión regular.

TABLA 3: LIMITADORES Y CUANTIFICADORES (REGEX)

Caracteres	Definición
^	Inicio de una línea.
$	Final de una línea.
*****	Aparece cero o más veces.
+	Aparece una o más veces.
{n}	Aparece exactamente n veces.
{n,}	Aparece al menos n veces.
{n,m}	Aparece al menos n veces y como máximo m veces.

Clases de caracteres y construcciones especiales

Define los caracteres que se aceptan y algunas construcciones especiales y caracteres especiales para evaluar las expresiones regulares.

TABLA 4: CLASES DE CARACTERES Y CONSTRUCCIONES ESPECIALES (REGEX)

Caracteres	Definición
[abc]	Se aceptan los caracteres a, b o c.
[^abc]	Cualquier carácter excepto a, b o c.
[a-z]	Caracteres desde a hasta z. También es aplicable para dígitos [0-9] y los caracteres en letras mayúsculas [A-Z].
.	Cualquier carácter.
\d	Un dígito del [0-9].
\s	Un carácter de espacio como pueden ser los siguientes [\t\n\x0B\f\r]. El primer carácter es un espacio en blanco " ".
\w	Un carácter de palabra [a-zA-Z_0-9]
\D	Cualquier no dígito [^0-9].
\S	Cualquier carácter que no sea de espaciado [^\s].
\W	Cualquier carácter que no sea un carácter de palabra [^\w].
****	Es un carácter para el escapado de caracteres. Es decir \\ en realidad se refiere a "\", o \{ se refiere a "{".
(?=)	Comprobación positiva a priori. Esto significa que se comprueba lo que siga al =, en caso de que se encuentre una coincidencia se procede a comprobar la siguiente parte de la expresión regular.

Una vez se conocen los caracteres que emplean las expresiones regulares en Java, se procede desarrollar métodos que emplean expresiones regulares para finalmente poder generar unas *whitelist* y *blacklist* propias.

Whitelists

Se comenzará por los métodos que permitirán que se desarrolle una *whitelist*. Este primer caso es muy sencillo y simplemente comprobará que el input contiene letras aceptadas dentro del sistema.

```java
public static boolean hasLetters(String s){
    boolean res = false;
    //Comprobar si tiene 1 o mas letras de a-z y A-Z
    Pattern p = Pattern.compile("^[a-zA-Z]+$");
    Matcher m = p.matcher(s);
    if(m.matches()){
        res = true;
    }
    return res;
}
```

Figura 60: Código del método hasLetters

Como se puede observar este ejemplo es bastante simple, dado que se comprueba que el input empiece o acabe por una o más letras en el dominio de la [a-zA-Z], es decir, permite tanto minúsculas como mayúsculas.

El siguiente ejemplo incorpora que la expresión regular acepte dígitos, caracteres españoles y el carácter "_". Este ejemplo se puede tomar como un ejemplo básico de *whitelist* para los caracteres aceptados en un sistema.

```java
public static boolean whiteListCharacters(String s){
    boolean res = false;
    //Similar a "^\\w+$", pero en este caso no se tendrían en cuenta los caracteres españoles
    Pattern p = Pattern.compile("^[a-zA-Z_0-9ñÑáéíóúÁÉÍÓÚ]+$");
    Matcher m = p.matcher(s);
    if(m.matches()){
        res = true;
    }
    return res;
}
```

Figura 61: Whitelist de caracteres

Cabe destacar que esta expresión regular se podría modificar por la clase predefinida \w que tiene el dominio de [a-zA-Z_0-9]. Aun así, es recomendable definir todos los caracteres aceptados para ser consistentes con la definición de *whitelist*. Adicionalmente, tal y como se menciona en el comentario la clase predefinida \w no tendría en cuenta los caracteres españoles como la ñ y las vocales con tilde.

El siguiente ejemplo es una propuesta de un método para emplear como *whitelist* que compruebe contraseñas. En este caso se ha tenido en cuenta que el proceso de registro siga una política de creación de contraseñas en la que se exija que la contraseña contenga una letra minúscula, una letra mayúscula, un dígito, un mínimo de 8 caracteres y un máximo de 160 caracteres. El límite inferior de caracteres se ha escogido conforme a las reglas del NIST en esta publicación [14] en el apartado 5.1.1.1

```java
//Whitelist check for passwords
/*True --> follows password policy
False --> does NOT follow password policy
*/
public static boolean passwordCheck(String password){
    boolean res = false;
    /*Explicacion del patron
     * (?=.*[a-z]+) --> contenga una o mas letras minusculas
     * (?=.*[A-Z]+) --> contenga una o mas letras mayusculas
     * (?=.*[0-9]+) --> contenga uno o mas digitos
     * [a-zA-Z_0-9]{8,160} --> sea una palabra de longitud minima 8 y maxima 160 con los caracteres [a-zA-Z_0-9]
     *
     * */
    Pattern p = Pattern.compile("(?=.*[a-z]+)(?=.*[A-Z]+)(?=.*[0-9]+)[a-zA-Z_0-9]{8,160}");
    /*En caso de seguir con el patron español,
     * la regex quedaría: "(?=.*[a-záéíóúñ]+)(?=.*[A-ZÁÉÍÓÚÑ]+)(?=.*[0-9]+)[a-zA-Z_0-9áéíóúñÁÉÍÓÚÑ]{8,160}"*/
    Matcher m = p.matcher(password);
    if(m.matches()){
        res = true;
    }
    return res;
}
```

Figura 62: Método para comprobar la politica de contraseña

En este caso se realizará una explicación más específica de cómo se ha conseguido comprobar que la expresión regular tenga una o más minúsculas, una o más mayúsculas y uno o más dígitos. Con tal de no repetir conceptos se explicará para el caso de las letras minúsculas, pero la explicación aplica para los otros dos casos restantes.

La expresión (?=.*[a-z]+) realiza una comprobación positiva, es decir, evalúa la expresión pero al finalizar se vuelve al punto inicial para dejar que la siguiente parte de la expresión regular evalúe todo el input. Dentro de la expresión (?=.*[a-z]+) el ".*" significa que busque cero o más caracteres, seguido de "[a-z]+" que significa que aparezca una o más de las letras en el dominio de la a hasta la z. La expresión regular estaría realizando entonces el siguiente proceso "comprueba a priori si existe cero o más caracteres de cualquier tipo y que tenga seguido una o más letras minúsculas del dominio a-z".

Llegado este punto, se ha podido observar cómo generar *whitelists* simples que cumplen con el objetivo de comprobar que los caracteres del input sea el especificado en la expresión regular.

Como ejercicio adicional se propone un método que permita verificar correos electrónicos empleando expresiones regulares. Previa a la demostración de los métodos para verificar los correos electrónicos es necesario saber cuál es la estructura que tienen, los caracteres permitidos y la longitud de los campos. El correo electrónico está compuesto por los campos *local-part@domain* según está definido en la RFC5321 [66]. La longitud de dichos campos está definida también en la RFC5321 en la sección 4.5.3, en la que se especifica que el campo *local-part* puede tener hasta 64 caracteres y el campo *domain* puede tener hasta 255 caracteres. En la RFC5321 se hace referencia a octetos y no caracteres para no limitarse a caracteres que ocupen un octeto.

Teniendo en cuenta la estructura del correo electrónico, la longitud total del correo electrónico sería de 64+1+255 = 320. El problema yace en que en el propio RFC5321 en la sección 4.5.3.1.3 se especifica que la longitud máxima del *Path* es de 256 caracteres. Teniendo en cuenta que el *Path* tendría la siguiente forma: "<"*Path*">", los caracteres efectivos serían en realidad de 254 caracteres para una dirección de correo electrónico. Adicionalmente, la longitud máxima de los TLD (*Top Level Domain*), que son los dominios que se encuentran en la parte superior de la jerarquía de los DNS (*Domain Name System*), según la RFC1034 [67] es de 63 caracteres.

Una vez se han explicado los estándares que se siguen para la formación de un correo electrónico, se procede a mostrar los distintos métodos para validar que el input sea un correo electrónico. Evidentemente, tener en cuenta todas las restricciones empleando solamente

expresiones regulares sería difícil y generaría una expresión regular enorme. Ésta última opción ya existe para Perl [68]. Debido a esto, se intentará buscar un término medio que facilite el registro de la mayoría de los correos electrónicos teniendo en cuenta las directrices de las RFC correspondientes.

El primer acercamiento sería el siguiente:

```
public static boolean basicEmailCheck(String email){
    boolean res = false;
    Pattern p = Pattern.compile("^[a-zA-Z_0-9.%+-]+@[a-zA-Z_0-9.-]+.[a-zA-Z]{2,}$");
    Matcher m = p.matcher(email);
    if(m.matches()){
        res = true;
    }
    return res;
}
```

Figura 63: Método para comprobar emails (básico)

En esta expresión regular, se comprueba que los caracteres en el *local-part* sean los correctos, luego se espera el separador "@" seguido del *domain* que generalmente tiene la pinta de "*subdomain.domain*".

El siguiente ejemplo mejora el previo teniendo en cuenta las restricciones del tamaño del correo electrónico.

```
public static boolean emailCheck(String email){
    boolean res = false;
    Pattern p = Pattern
            .compile("^[a-zA-Z0-9!#$%&'*+-/=?^_`{|}~][a-zA-Z0-9!#$%&'*+-/=?^_`{|}~.]{1,63}@"
                    + "[a-zA-Z0-9-]{1,125}.[a-z]{2,63}$");
    Matcher m = p.matcher(email);
    if(m.matches()){
        res = true;
    }
    return res;
}
```

Figura 64: Método para comprobar emails (avanzado)

Se puede apreciar que en esta expresión regular se añaden ciertos caracteres que pueden aparecen en las distintas partes del correo electrónico que no se tuvieron en cuenta en el ejemplo anterior además de añadir las restricciones de longitud. La longitud de los campos se ha escogido en base al siguiente razonamiento.

Teniendo en cuenta que la estructura común de un correo electrónico es *local-part@subdomain.domain*, el campo de *local-part* puede tener un máximo de 64 caracteres. Esta primera comprobación se realiza definiendo dos dominios, el primero hace referencia al primer carácter que no puede ser un punto y en el segundo dominio se incluye el "." para permitir correos del estilo "jose.manuel". Acto seguido se define el separador "@" para distinguir la *local-part* del *domain*. El campo *domain,* al ser generalmente *subdomain.domain* y teniendo en cuenta que el TLD tiene como máximo 63 caracteres, se le asigna primero la longitud al *domain* que sigue al ".". En este momento se tiene un total de 64+1+1+63 = 129 caracteres de los 254 que se permiten. Por tanto, el campo restante que es *subdomain* se le adjudica una longitud máxima de 125 caracteres.

Este ejemplo tampoco es perfecto porque algunos de los caracteres que se permiten en la *local-part* están sujetos a condiciones más restrictivas, pero en este caso se permite que puedan ocurrir. La decisión de hasta qué punto se añaden más reglas y se complica la validación del correo electrónico es algo que se debe evaluar. Una posible mejora sería permitir que el campo

subdomain pueda tener el carácter ".", para permitir direcciones como "jmgosupport@support.service.com".

Blacklists

A continuación, se mostrará un ejemplo de una *blacklist* con un conjunto de caracteres que posiblemente no deban ser aceptados en el sistema.

```
public static boolean characterBlacklist(String s){
    boolean res = false;
    Pattern p = Pattern.compile("[^<>%^$/;\"]+");
    Matcher m = p.matcher(s);
    if(m.matches()){
        res = true;
    }
    return res;
}
```

Figura 65: Blacklist de caracteres

Como se puede apreciar es la negación de los caracteres "<>%^$/", de esta forma en caso de que el método devuelva *false,* significará que el input introducido contiene alguno de los símbolos de la *blacklist* y debería descartarse. En el caso de que no aparezcan los símbolos de la *blacklist*, el resultado obtenido será *true.*

Por último, se ha de mencionar que el uso de las *whitelist* está recomendado sobre el uso de las *blacklist* en la página de OWASP [69]. La razón de esto es que las *whitelist*, como se ha mencionado al inicio del ejercicio definen lo que está permitido y por definición todo lo demás no lo está. En el caso contrario se encuentran las *blacklists,* que definen lo que no está permitido. Desde un punto de vista de seguridad las *blacklists* ayudan a la prevención de ataques conocidos, mientras que las *whitelists* en principio pueden cubrir casos de ataque aún desconocidos. Esto no quiere decir que no se deban emplear las *blacklists,* y que simplemente se empleen las *whitelists*, se pueden emplear ambos mecanismos para asegurar que se reciben datos permitidos, a través de las *whitelists*, y se vuelve a confirmar mediante las *blacklists* que los datos no puedan ocasionar un ataque conocido.

4.3. Comprobación de la eficacia de los Materiales Didácticos.

En este apartado se cubrirá la validación necesaria para saber si los materiales didácticos le han sido de utilidad a los alumnos y les han aclarado conceptos y mecanismos mediante los que pueden proteger las credenciales de usuario. Para ello se han generado las siguientes cuestiones a las que los alumnos pueden responder para validar sus conocimientos adquiridos. Las cuestiones son las siguientes:

1. Enumera tres ataques que se pueden realizar a los hash *digest*, para obtener el valor original de la contraseña. (1pto)
2. ¿Qué es lo que contiene una *wordlist*? (0,5pto)
3. ¿Cuál es la razón por la que se crea un usuario adicional para administrar la base de datos en vez de emplear el rol por defecto? (0,75pto)
4. Explica el sistema de permisos de archivos en sistemas Unix. ¿Cómo puede modificarse el propietario de un archivo? (0,75pto)
5. ¿Qué propiedades deben cumplir las funciones hash para ser consideradas seguras frente a ataques? Explica brevemente cada una de ellas. (1pto)
6. ¿Con qué motivo se usan los *salts* en el proceso de hashear contraseñas?, ¿Cuál es el mecanismo que se emplea para generar *salts* criptográficamente seguros? (0,75pto)
7. ¿Cuántas veces se debe emplear un *salt* generado? ¿Cuántos usuarios pueden compartir un *salt*? ¿Cómo se debe almacenar el *salt* asociado a un hash? (1pto)
8. Nombra dos funciones hash que no sean seguras contra ataques. Nombra a tres funciones consideradas seguras. (1pto)
9. ¿Qué tipos de cifrado existen? Explica brevemente el funcionamiento de cada uno junto a sus ventajas e inconvenientes. Enumera tres tipos de modo de operación de cifradores de bloque. (2,25pto)
10. Describe los conceptos de *whitelist* y *blacklist*. ¿De qué tipo de ataques permite proteger cada una? (1pto)

Para que el alumno supere la prueba de validación es necesario que obtenga una nota total de 5/10 demostrando así que ha adquirido unos conocimientos mínimos respecto a la protección de credenciales de usuario y los ataques existentes.

A continuación, se muestran las posibles soluciones que se pueden presentar para obtener los puntos de cada pregunta presente en la prueba de validación.

1. **Enumera tres ataques que se pueden realizar a los hash *digest*, para obtener el valor original de la contraseña. (1pto)**
   Ataque de fuerza bruta, de diccionario, ataque mediante una *lookup table*, *reverse lookup table* o *rainbow table.*
2. **¿Qué es lo que contiene una *wordlist*? (0,5pto)**
   Una *wordlist* contiene un conjunto de palabras que son candidatas a ser potenciales contraseñas, suelen generarse a partir de las contraseñas rescatadas de *password dump sites* a las que se le aplican reglas para derivar más palabras.
3. **¿Cuál es la razón por la que se crea un usuario adicional para administrar la base de datos en vez de emplear el rol por defecto? (0,75pto)**
   Se crea para evitar el uso del usuario administrador de toda la base de datos que generalmente suele venir por defecto a root-root, entiéndase los campos como nombre-contraseña, y así limitar la capacidad de operabilidad de un atacante en caso de tomar posesión de la cuenta.

4. **Explica el sistema de permisos de archivos en sistemas Unix. ¿Cómo puede modificarse el propietario de un archivo? (0,75pto)**

   Cada fichero y directorio tiene una serie de permisos que están divididos en tres secciones, usuario, grupo y el mundo. Cada sección tiene para otorgar tres permisos, que son de lectura, escritura y ejecución. Los permisos se representan mediante las letras *r*,*w* y *x* para los permisos de lectura, escritura y ejecución respectivamente. Para modificar el usuario propietario se puede hacer uso del comando *chmod XYZ <path-fichero>* donde X, Y, Z son la representación octal de los permisos adjudicados, el valor máximo es 7 y el mínimo el 0.

5. **¿Qué propiedades deben cumplir las funciones hash para ser consideradas seguras frente a ataques? Explica brevemente cada una de ellas. (1pto)**

   Las funciones hash deben cumplir con las siguientes propiedades para ser consideradas seguras ante ataques:

   - **Resistencia a pre-imágenes:** dado un valor de hash *h* debería ser difícil encontrar otro mensaje o input *m* de forma que *h = hash(m)*. Esta propiedad hace referencia a la cualidad de una función de una sola dirección, de forma que teniendo solamente el hash resultante es imposible generar el input del que procede.
   - **Resistencia a segunda pre-imagen:** dado un input m_1, debería ser difícil encontrar otro input distinto m_2 de tal forma que hash(m_1) = hash(m_2). En el caso de las credenciales de usuario, se refiere a que, si se obtiene el hash de la contraseña, un input incorrecto que no sea la contraseña no debe generar el mismo hash.
   - **Resistencia a colisiones:** debería ser suficientemente difícil encontrar dos mensajes m_1 y m_2, siendo $m_1 \neq m_2$, tal que hash(m_1) = hash(m_2). Este par, en caso de encontrarse, se llama una colisión.

6. **¿Con qué motivo se usan los *salts* en el proceso de hashear contraseñas?, ¿Cuál es el mecanismo que se emplea para generar *salts* criptográficamente seguros? (0,75pto)**

   El uso de los *salt* criptográficos tiene el objetivo de añadir aleatoriedad al input para que el hash no sea predecible y las técnicas TMTO no sean efectivas. El mecanismo para generar *salts* criptográficamente seguros es mediante el uso de los CSPRNG (*Cryptographically secure pseudorandom number generator*) que se aseguran de obtener un número con un nivel de entropía seguro.

7. **¿Cuántas veces se debe emplear un *salt* generado? ¿Cuántos usuarios pueden compartir un *salt*? ¿Cómo se debe almacenar el *salt* asociado a un hash? (1pto)**

   Un *salt* se debe emplear una única vez. No se debe compartir un *salt* para más de un usuario. El valor del *salt* se almacena en claro para poder realizar el proceso de autentificación, es posible cifrarlo, pero no necesario.

8. **Nombra dos funciones hash que no sean seguras contra ataques. Nombra a tres funciones consideradas seguras. (1pto)**

   Las funciones hash más conocidas que no son seguras contra ataques hash son MD5 y SHA1. Las funciones hash consideradas seguras actualmente son *bcrypt, scrypt, PBKDF2 y Argon2.*

9. **¿Qué tipos de cifrado existen? Explica brevemente el funcionamiento de cada uno junto a sus ventajas e inconvenientes. Enumera tres tipos de modo de operación de cifradores de bloque. (2,25pto)**

   El cifrado simétrico se basa en el uso de una única clave secreta que es conocida por las entidades responsables del cifrado y descifrado, permanece desconocida para el resto del mundo. El cifrado simétrico emplea la misma clave para cifrar y descifrar, por ello es

necesaria la protección de la clave. La principal ventaja del cifrado simétrico es la rapidez respecto al cifrado asimétrico, así como el uso de menos recursos del sistema.

El cifrado asimétrico también conocido como cifrado de clave pública es otro de los cifrados disponibles. Este cifrado se basa en el uso de dos claves, una de ellas se llama clave pública y como indica su nombre y es conocida por todo el mundo. La otra clave se llama clave privada y esta debe ser secreta y sólo ser conocida por el propietario del par de claves. El proceso de cifrado del sistema asimétrico se centra en el cifrado de los datos empleando la clave pública del destinatario y el descifrado de los mismos empleando la clave privada correspondiente. Las principales ventajas del cifrado asimétrico es que es bastante difícil de romper y bastante potente ya que se requieren las dos claves para el cifrado y descifrado. Los inconvenientes respecto al cifrado simétrico es que es bastante más lento y emplea muchos más recursos.

Existe otro tipo de cifrado que deriva del uso de los dos cifrados mencionados previamente. Este cifrado se llama cifrado híbrido y se encarga de emplear las ventajas del cifrado simétrico para poder cifrar los datos de forma rápida y sin gastar demasiados recursos del sistema mientras que gracias al sistema de cifrado asimétrico se mantiene a salvo la clave secreta empleada en el cifrado simétrico. De esta forma se cifran los datos con gran eficiencia y la clave simétrica está protegida por el cifrado asimétrico que es un sistema bastante robusto.

Los modos de operación de cifradores de bloque son los siguientes:

- *ECB, Electronic Code Book Mode*
- *CBC, Cipher Block Chaining Mode*
- *CFB, Cipher Feedback Mode*
- *OFB, Output Feedback Mode*
- *CTR, Counter Mode*

10. Describe los conceptos de *whitelist* y *blacklist*. ¿De qué tipo de ataques permite proteger cada una? (1pto)

Las *whitelist* definen lo que está permitido y por definición todo lo demás no lo está. En el caso contrario se encuentran las *blacklists,* que definen lo que no está permitido. Desde un punto de vista de seguridad las *blacklists* ayudan a la prevención de ataques conocidos, mientras que las *whitelists* en principio pueden cubrir casos de ataque aún desconocidos.

5. GESTIÓN DEL PROYECTO

5.1. Planificación del proyecto

En este apartado se explica la planificación llevada a cabo durante la realización del proyecto. En primer lugar, se expondrán las actividades de las que consta el proyecto desglosadas por categorías. Luego se comentan las fechas en las que se realizaron cada una de las actividades y una estimación de las horas dedicadas a lo largo de cada mes. Por último, se muestra un diagrama de Gantt en el que se puede visualizar la planificación del proyecto.

5.1.1. Desglose de actividades

El proyecto se divide en cuatro categorías principales, cada una de ellas está compuesta por subactividades. El desglose es el siguiente:

1. Búsqueda y Análisis de textos y bibliografía
    a. Planteamiento del problema
    b. Análisis de las técnicas actuales
    c. Propuesta de soluciones
2. Análisis de los TFG
    a. Análisis previo y filtrado de los TFG
    b. Análisis individual de los TFG
    c. Interpretación de los resultados
3. Diseño
    a. Diseño y elección de materiales didácticos
    b. Generación e implementación de los materiales didácticos
    c. Generación de la prueba de concepto
4. Gestión del proyecto
    a. Documentación
        i. Redacción del documento
        ii. Revisión del documento
    b. Entorno socioeconómico
    c. Marco regulador

La primera categoría se centra en la recopilación de textos y bibliografía y el estudio de los mismos para una mejor comprensión de los conceptos que se van a emplear a lo largo del proyecto. La segunda categoría se centra en el proceso de lectura y filtrado de los TFG, así como su análisis individual y extracción de resultados. La tercera se centra en el diseño e implementación de los distintos materiales didácticos que apoyarán a afianzar el conocimiento de las técnicas explicadas en el documento. La última categoría hace referencia al proceso de generación del documento junto con sus correspondientes revisiones además del estudio del entorno socioeconómico y marco regulador.

5.1.2. Asignación de horas y fechas

En este apartado se describen las horas empleadas y las fechas en las que se llevaron a cabo. El proyecto comenzó a inicios de febrero, pero por diversas razones se pospuso su realización a junio, las horas dedicadas durante el mes de febrero no se tendrán en cuenta ya que hubo que repetir el proceso de análisis y búsqueda de textos para aclarar conceptos teóricos. Por tanto, el proyecto se desarrolla entre los meses de junio, julio, agosto y septiembre. A continuación, se muestra una tabla con los días durante los que se trabajó en cada mes y una media de las horas dedicadas.

TABLA 5: DISTRIBUCIÓN DE HORAS MEDIAS CADA MES

Mes	Días	Horas/día
Junio	Desde el día 11 al 24	Dos horas
Julio	Desde el día 9 al 22	Dos horas
Agosto	Desde el día 6 al 12	Cuatro horas
	Desde el día 20 al 31	Cinco horas
Septiembre	Desde el día 1 al 24	Ocho horas

En el mes de junio se trabajaron un total de 13 días, en el mes de julio se trabajó durante 13 días. Durante el mes de agosto se trabajó en total 17 días divididos en una primera sección de 6 días y otra sección de 11 días. A lo largo del mes de septiembre se ha trabajado durante los 24 días.

TABLA 6: DISTRIBUCIÓN DE HORAS TOTALES POR MES

Mes	Días	Horas/día	Total horas
Junio	13	2	26
Julio	13	2	26
Agosto	6	4	24
	11	5	55
Septiembre	24	8	216
TOTAL			347

El número total de horas empleadas para la realización del proyecto tal y como se muestra en la tabla es de 347 horas.

5.1.3. Diagrama Gantt

En este apartado se muestra el diagrama Gantt que se ha generado para el proyecto. Se ha empleado la herramienta *Tom's Planner* [70] para su generación. El diagrama muestra las semanas de cada uno de los meses, siendo 1 la primera semana de enero y 52 la última semana de diciembre.

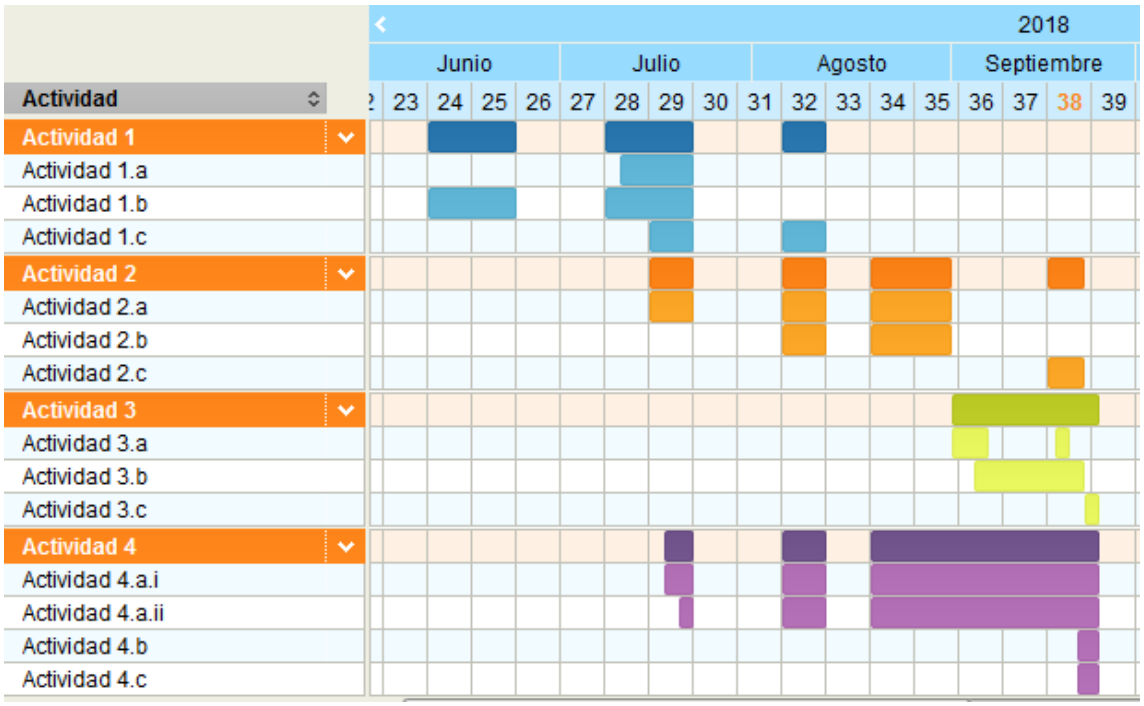


Figura 66: Diagrama de Gantt

6. ENTORNO SOCIOECONÓMICO Y MARCO REGULADOR

6.1. Presupuesto

En este apartado se realiza una estimación del presupuesto del desarrollo del proyecto en el que se tendrá en cuenta los gastos relacionados con los costes asociados al personal y los recursos utilizados.

6.1.1. Gastos de personal

En cuanto a los costes asociados al personal se ha tenido en cuenta que la lo largo del proyecto se ha desempeñado distintos roles. Los roles que se han adoptado a lo largo del proyecto son los siguientes:

- Jefe de proyecto, que se encarga de la dirección general del proyecto estableciendo los objetivos que se deben conseguir y reorientando el proyecto tras las revisiones.
- Analista, que se encarga de recabar información de las técnicas de protección de credenciales de usuario, así como el análisis individual de los TFG.
- Programador, que se encarga de la implementación del código necesario para la realización de los materiales didácticos.
- Documentador, es el encargado de generar el documento que recoge el trabajo realizado a lo largo del proyecto.

TABLA 7: COSTES EN GASTOS DE PERSONAL

Rol	Salario por horas (€/hora)	Horas dedicadas (horas)	TOTAL (€)
Jefe de proyecto	60,00	50	3.000,00
Analista	35,00	137	4.795,00
Programador	25,00	90	2.250,00
Documentador	20,00	70	1.400,00
TOTAL		347	11445,00

6.1.2. Gastos de los recursos utilizados

En este apartado se describen los gastos atribuidos a los recursos hardware, recursos software y material fungible empleados a lo largo del proyecto. En primer lugar, se describen los recursos hardware que se han necesitado para el proyecto.

TABLA 8: COSTES DE RECURSOS HARDWARE

Recursos Hardware	Coste (€)
Lenovo IdeaPad	780,00

A continuación, se muestra la tabla asociada al material fungible empleado a lo largo del proyecto.

TABLA 9: COSTES DE RECURSOS SOFTWARE

Recursos Software	Coste (€)
Microsoft Office 365	0,00 (*)
Licencia Windows 10	0,00 (*)
Kali Linux	0,00 (**)
VMware (versión gratuita)	0,00 (**)
John the Ripper (versión gratuita)	0,00 (**)
TOTAL	0,00

(*) La licencia del software provista de forma gratuita por la cuenta de alumno de la UC3M.

(**) Licencias de las herramientas gratuitas o versión de la comunidad

Por último, se muestran los gastos de los materiales fungibles empleados.

TABLA 10: COSTES DE MATERIAL FUNGIBLE

Material fungible	Coste (€)
Bolígrafos	5,00
Papeles	15,00
Carpetas	10,00
TOTAL	30,00

En base a esta tabla se calcula el total de los gastos de los recursos utilizados.

TABLA 11: COSTES TOTALES DE RECURSOS UTILIZADOS

Descripción	Coste (€)
Recursos Hardware	780,00
Recursos Software	0,00
Material Fungible	30,00
TOTAL	810,00

6.1.3. Gastos totales

En este apartado se muestran los costes totales del proyecto teniendo en cuenta los gastos de personal y recursos empleados. En cuanto a los gastos relativos a los recursos utilizados, refiriéndose a los hardware, software y fungibles, se aplicará un 21% de IVA. En el caso de los gastos de personal se aplicará un 24% adicional debido al IRPF y Seguridad Social

TABLA 12: GASTOS TOTALES DEL PROYECTO

Descripción	Coste sin IVA/IRPF (€)	Coste del IVA/IRPF (€)	TOTAL
Gastos de personal	11.445,00	2.746,80	14.191,80
Gastos de recursos utilizados	810,00	170,10	980,10
TOTAL	12.255,00	2.916,90	15.171,90

El presupuesto final del proyecto teniendo en cuenta los gastos de personal y gastos de recursos utilizados ascienden a un total de 15.171,90€.

6.2. Impacto socioeconómico

En este apartado se analiza cómo puede afectar el proyecto a la sociedad y si tiene un impacto económico. Respecto al impacto social, se espera que afecte a la audiencia objetivo que son los alumnos de ingeniería informática principalmente, pero también a cualquier otro trabajador en el campo de las tecnologías de la información. El efecto que tendrá en los alumnos es un aumento en la concienciación con relación a la protección de las credenciales de usuarios en sistemas informáticos. Debido a este aumento de concienciación se espera que los alumnos tengan en cuenta la implementación de mecanismos de seguridad de forma que la confidencialidad de los datos de los usuarios se puede asegurar. Esto afecta de forma directa a los futuro usuarios de los sistemas informáticos en los que se espera que los alumnos de las universidades trabajen, ya que, es bastante probable que dichos sistemas se vean afectados por los conocimientos de seguridad de los alumnos y se puedan implementar mecanismos de seguridad.

En cuanto al ámbito académico se ha conseguido realizar un estudio de la concienciación de los alumnos de la UC3M con respecto a su capacidad para proteger las credenciales de usuario y dados los resultados se espera que se tomen medidas para que los alumnos deban cubrir un mínimo en conocimientos de seguridad a la hora de desarrollar el sistema. De esta forma es probable que se valore a lo largo del grado, específicamente en ingeniería informática, el hecho de tener en cuenta la seguridad informática en asignaturas que no tengan que ser dedicadas a la misma.

Adicionalmente, en caso de que el lector sea alguien que no tenga relación con el desarrollo de sistemas informáticos a lo largo del proyecto se demuestra la necesidad de emplear contraseñas complejas o al menos no comunes y mucho menos mantener la misma contraseña para todos los servicios informáticos. En este sentido el documento puede servir como documento informativo para ser más conscientes a la hora de escoger contraseñas.

Respecto al posible impacto económico no se espera que el proyecto como tal tenga una influencia fuerte ya que está más orientado a un documento de investigación y no al desarrollo de un sistema que se pueda implantar en el mercado. Es cierto, que si el proyecto tiene éxito sería factible que la seguridad se tenga más en cuenta en las empresas y por tanto en caso de que los sistemas de las empresas se vean comprometidos es menos probable que se produzcan perdidas de datos personales. Es importante recordar que la confianza de los usuarios en el sistema es un factor importante y una pérdida de datos personales de los usuarios puede suponer una pérdida masiva de usuarios.

6.3. Marco Regulador

En este apartado se realiza un estudio de las legislaciones aplicables al desarrollo de este trabajo, en específico a los materiales didácticos generados. Adicionalmente, se hará un estudio de los estándares técnicos que se deben cumplir.

6.3.1. Legislación aplicable

En primer lugar, se comentará la no aplicación de la LOPD y RGPD en este TFG puesto que a la hora de obtener los TFG de los alumnos de la UC3M se han recabado parte de sus datos personales como es el nombre completo. El caso es que los datos obtenidos de los autores están publicados en el repositorio de e-Archivo [17] y pueden ser accedidos de forma pública debido a que el repositorio se inscribe en el movimiento *Open Access Initiative* y todos los autores de los TFG publicados han debido aceptar previamente la publicación de acceso gratuito a su trabajo. Ya que este TFG se añadirá en un futuro al mismo repositorio todos los datos que aparecen aquí recabados serán públicos y se podrán recabar de la misma forma que se ha realizado para este proyecto.

El resto de las legislaciones aplicables a este proyecto derivan del apoyo de creaciones de terceros para generar los materiales didácticos. En primer lugar, la herramienta *John the Ripper* [4] que se ha empleado para el primer material didáctico. Esta herramienta está publicada, en su versión gratuita, bajo la licencia GNU GPL v2 o posterior. Es necesario aclarar que no se trata de la licencia GNU GPL v3 ya que es más restrictiva, esto está descrito en la página oficial. En el caso del material didáctico se ha hecho uso de la versión mejorada por la comunidad, aunque esta referencia a la licencia empleada en la página oficial.

La siguiente licencia que se debe tener en cuenta también es GNU GPL v2 o posterior debido al uso del proyecto *rexgen* [49] a la hora de hacer la instalación propia de la herramienta *John the Ripper*.

Otra de las licencias que afectan al proyecto es la licencia CC0 [71] ya que en el tercer material didáctico se hace una introducción al uso de Argon2. El proyecto original de los creadores del algoritmo además de estar bajo la licencia CC0, se encuentra bajo la licencia Apache 2.0 License [72]. En cuanto al proyecto empleado para el uso de Argon2 en Java, está establecido bajo una licencia de LGPL v3.

Por último, hay que mencionar que la licencia de Kali Linux empleado en el primer material didáctico está bajo una licencia GNU GPL v3.

Todas las licencias que se han empleado a la hora de desarrollar el proyecto y llevar a cabo los materiales didácticos son compatibles y por tanto es software libre, la compatibilidad se puede comprobar en la página oficial [73].

6.3.2. Estándares técnicos

Durante el desarrollo del proyecto se ha tenido que tener en cuenta varios estándares para la generación correcta de los materiales didácticos. El conjunto al que más referencia se ha hecho es el de las RFC. Una RFC (*Request for Comments*) es un documento formal del IETF (*Internet Engineering Task Force*) que es el resultado de varias revisiones. Algunos RFC son informativos, pero otros funcionan como estándares de internet. Entre las distintas RFC mencionadas en el proyecto se encuentran la RFC2104, RFC5321, RFC1034 y el nuevo borrador para la RFC sobre Argon2.

El otro conjunto de estándares que se ha tenido en cuenta a lo largo del proyecto pertenece al NIST (*National Institute of Standards and Technology*). Varias de estas referencias son a artículos y pautas que aporta el NIST para el uso de funciones hash, identificación digital y modos de operación en cifradores de bloque.

7. CONCLUSIONES

7.1. Conclusiones y Objetivos cumplidos

Una vez terminado el proyecto, se analiza los objetivos que se han cumplido y se extraen unas conclusiones sobre el trabajo desarrollado.

En el documento se ha podido ofrecer una vista general de las técnicas que actualmente son más empleadas para la protección de las credenciales de usuario. Además, se han introducido conceptos de seguridad básicos para que cualquier lector que no tenga conocimientos previos de seguridad informática sea capaz de comprender los conceptos explicados. Adicionalmente, se ha conseguido realizar una propuesta de soluciones basada en las técnicas actuales, específicamente en aquellas que añaden un grado de seguridad mínimo para que las credenciales son sean usurpadas con facilidad, pero sin tener que poner una carga excesiva en los siguientes autores de los futuros TFG para implementarlas.

Aparte se ha llevado a cabo un estudio sobre el grado de seguridad respecto a las credenciales de usuario en algunos TFG recientes. Cada uno de esos TFG ha sido analizado de forma individual lo que ha permitido una mayor comprensión del proyecto en cuestión. En base a dichos análisis se ha llevado a cabo una evaluación de los datos obtenidos a partir de los cuales se ha propuesto varias hipótesis que den explicación al comportamiento de las subidas y bajadas del nivel de seguridad a lo largo de los 4 años evaluados. Incluso, se ha expuesto algunas de las estrategias más comunes llevadas a cabo por los alumnos a la hora de tener que concebir la seguridad en su sistema. Debido a esto, se ha sido capaz de vislumbrar con mayor facilidad los problemas que tienen los alumnos a la hora de incluir la confidencialidad de las credenciales de usuario en sus aplicaciones.

En base a las técnicas estudiadas, las propuestas y la identificación de los problemas de los autores al enfrentarse a la seguridad informática, se han generado una serie de materiales didácticos que consiguen recoger todas las soluciones propuestas, teniendo en cuenta las tecnologías más empleadas por los usuarios.

7.2. Líneas futuras de trabajo

El proyecto tiene varios puntos que pueden mejorarse y volver a evaluar en trabajos futuros. Esto se debe a que la seguridad informática es un campo que va mejorando cuando se hacen ataques y descubren vulnerabilidades sobre los métodos empleados actualmente. Por ello, es necesario que el proyecto revise los algoritmos propuestos con los que estén en vigor en el momento de la revisión para saber si efectivamente se están aportando algoritmos seguros o ya están rotos y no son útiles criptográficamente.

Haciendo referencia a la parte del estudio llevado a cabo en el proyecto, sería interesante investigar y analizar la seguridad de otros TFG más antiguos para ver si existe algún patrón o ver qué acontecimientos marcan las diferencias en la seguridad. Adicionalmente, como es lógico el estudio se podría llevar a cabo en un plazo de cuatro o cinco años para comprobar si efectivamente el proyecto ha tenido impacto y ha ayudado a los alumnos a mejorar el grado de seguridad o si por el contrario no ha sido exitoso. En el caso que se vuelva a llevar a cabo otro estudio también sería interesante aumentar el número de muestras recogidas para poder tener unas estadísticas más significativas y extraer conclusiones sobre ellas. Finalmente, en el nuevo

estudio realizado sería necesario la comprobación de la RGPD y no tener en cuenta solamente a la LOPD en cuanto a las leyes que deberían afectar al proyecto analizado.

Otra posibilidad es que, en vez de mantener el proyecto centrado en la protección de las credenciales de usuario, se haga un estudio sobre otros campos de la seguridad informática en el proyecto, como puede ser protocolos de comunicación empleados y protección frente a ataques conocidos.

Respecto a los materiales didácticos sería una gran mejora el aportar material audiovisual como pueden ser tutoriales con una duración aproximada de diez minutos en los que se expliquen los conceptos teóricos del ejercicio y se lleve a cabo un ejemplo guiado. En adición a estos nuevos materiales multimedia, se podría además generar una serie de ejercicios adicionales para los alumnos de forma que puedan practicar los conceptos por sí mismos, en vez de seguir el ejercicio guiado.

Otra posible inclusión sería añadir un estudio de las distintas leyes o estándares que rigen la seguridad y hacer un resumen informativo sobre ellas que contenga los puntos más importantes.

8. BIBLIOGRAFÍA

[1] Parlamento Europeo y Consejo de la Unión Europea, "REGLAMENTO (UE) 2016/679 DEL PARLAMENTO EUROPEO Y DEL CONSEJO de 27 de abril de 2016 relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/46/CE (Reglamentogeneral de protección de datos)", Boe.es, 27-04-2016. **[En Linea] Disponible en:** https://www.boe.es/doue/2016/119/L00001-00088.pdf. [Último acceso: 25/09/2018]

[2] worawita," Bruter", SourceForge**,** 07-08-2015**. [En línea]. Disponible en:** https://sourceforge.net/projects/worawita/. [Último acceso: 25/09/2018]

[3] vanhauser-thc**," thc**-hydra", *Github***. [En línea]. Disponible en:** https://github.com/vanhauser-thc/thc-hydra/. [Último acceso: 25/09/2018]

[4] Openwall, "John the Ripper password cracker", *Openwall*. **[En línea]. Disponible en:** https://www.openwall.com/john/. [Último acceso: 25/09/2018]

[5] darknet, "Brutus Password Cracker - Download brutus-aet2.zip AET2", *Darknet*, 2017. **[En línea]. Disponible en:** https://www.darknet.org.uk/2006/09/brutus-password-cracker-download-brutus-aet2zip-aet2/. [Último acceso: 25/09/2018]

[6] Ron Bowes, "Passwords - SkullSecurity", *Wiki.skullsecurity.org*, 2015. **[En línea]. Disponible en:** https://wiki.skullsecurity.org/Passwords. [Último acceso: 25/09/2018]

[7] R. JAŠEK, L. SARGA and R. BENDA, *Wseas.us*, 2013. **[En línea]. Disponible en:** http://www.wseas.us/e-library/conferences/2013/Valencia/ACIC/ACIC-01.pdf. [Último acceso: 25/09/2018]

[8] NIST, "National Institute of Standards and Technology | NIST", *NIST*. **[En línea]. Disponible en:** https://www.nist.gov/. [Último acceso: 25/09/2018]

[9] NIST, "NIST Policy on Hash Functions - Hash Functions | CSRC", *Csrc.nist.gov*. **[En línea]. Disponible en:** https://csrc.nist.gov/Projects/Hash-Functions/NIST-Policy-on-Hash-Functions. [Último acceso: 25/09/2018]

[10]"SHAttered", *Shattered.io*. **[En línea]. Disponible en:** https://shattered.io/. [Último acceso: 25/09/2018]

[11]P. Oechslin, "LASEC", *Lasec.epfl.ch*. **[En línea]. Disponible en:** https://lasec.epfl.ch/php_code/publications/search.php?ref=Oech03. [Último acceso: 25/09/2018]

[12]"Password Hashing Competition", *Password-hashing.net*. **[En línea]. Disponible en:** https://password-hashing.net. [Último acceso: 25/09/2018]

[13] H Krawczyk, "HMAC: Keyed-Hashing for Message Authentication ", *Ietf.org*. **[En línea]. Disponible en:** https://www.ietf.org/rfc/rfc2104.txt. [Último acceso: 25/09/2018]

[14] NIST, "NIST Special Publication 800-63B", *Pages.nist.gov*. **[En línea]. Disponible en:** https://pages.nist.gov/800-63-3/sp800-63b.html. [Último acceso: 25/09/2018]

[15] NIST, "Recommendation for Block Cipher Modes of Operation. Methods and Techniques", *Nvlpubs.nist.gov*. **[En línea]. Disponible en:**

https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38a.pdf. [Último acceso: 25/09/2018]

[16] UC3M, "Inicio | UC3M", *Uc3m.es*. **[En línea]. Disponible en:** https://www.uc3m.es/Inicio. [Último acceso: 25/09/2018]

[17] e-Archivo, "e-Archivo Principal", *E-archivo.uc3m.es*. **[En línea]. Disponible en:** https://e-archivo.uc3m.es/. [Último acceso: 25/09/2018]

[18] H. Mata San Juan**, "**Herramienta de análisis de legibilidad de contenidos educativos **", Trabajo fin de grado,** Departamento de Informática**,** Universidad Carlos III de Madrid**,** Madrid**,** España**,** 2016**. [En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/24301.

[19] A. Bernárdez Martínez**, "**Modelado de un puerto marítimo con Unity 3D**", Trabajo fin de grado,** Departamento de Informática**,** Universidad Carlos III de Madrid**,** Madrid**,** España**,** 2016**. [En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/24302.

[20] C.E. Villegas Flores**, "** Desarrollo de una aplicación multimodal para apoyar el estudio mediante m-learning**", Trabajo fin de grado,** Departamento de Informática**,** Universidad Carlos III de Madrid**,** Madrid**,** España**,** 2017**. [En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/27246.

[21] J. Valdés Fernández**, "** Aplicación nutricional y seguimiento deportivo, en atletas de alto rendimiento**", Trabajo fin de grado,** Departamento de Informática**,** Universidad Carlos III de Madrid**,** Madrid**,** España**,** 2016**. [En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/24299.

[22] V. Labrado Vicente**, "** Aplicación móvil para la detección de pasos basada en acelerometría a baja velocidad**", Trabajo fin de grado,** Departamento de Informática**,** Universidad Carlos III de Madrid**,** Madrid**,** España**,** 2017**. [En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/27279.

[23] V. M. Ramírez Marcos**, "** Diseño e implementación de un sistema de control remoto para vigilancia en Android", **Trabajo fin de grado,** Departamento de Informática**,** Universidad Carlos III de Madrid**,** Madrid**,** España**,** 2017**. [En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/27278.

[24] A. Rivero Ortíz**, "** Desarrollo de gestor de comidas a domicilio mediante la aplicación de mensajería instantánea Telegram**", Trabajo fin de grado,** Departamento de Informática**,** Universidad Carlos III de Madrid**,** Madrid**,** España**,** 2016**. [En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/24290.

[25] A. Muñoz Moreno**, "** Desarrollo de un asistente multimodal educativo para dispositivos móviles Android**", Trabajo fin de grado,** Departamento de Informática**,** Universidad Carlos III de Madrid**,** Madrid**,** España**,** 2016**. [En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/24289.

[26]H. Santos Senra**, "** Diseño e implementación de un sistema domótico basado en Raspberry Pi**", Trabajo fin de grado,** Departamento de Informática**,** Universidad Carlos III de Madrid**,** Madrid**,** España**,** 2017**. [En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/26313.

[27] J. Vázquez Coll, " Simulación de trayectorias de barcos y aplicación al control marítimo", **Trabajo fin de grado,** Departamento de Informática, Universidad Carlos III de Madrid, Madrid, España, 2017. **[En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/23709.

[28] C. E. Pérez Moscoso, " Estrategias de diversificación eficiente de carteras e implementación de una plataforma digital de inversión", **Trabajo fin de grado,** Departamento de Informática, Universidad Carlos III de Madrid, Madrid, España, 2015. **[En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/23695.

[29] M. San José de Vicente, " Desarrollo de una aplicación de turismo gastronómico para dispositivos iOS y análisis estadístico ", **Trabajo fin de grado,** Departamento de Informática, Universidad Carlos III de Madrid, Madrid, España, 2016. **[En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/23671.

[30] D. Sánchez Muñoz, " Diseño e implementación de Campus Life, una aplicación móvil para la consulta e inscripción de actividades deportivas y culturales", **Trabajo fin de grado,** Departamento de Informática, Universidad Carlos III de Madrid, Madrid, España, 2016. **[En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/27096.

[31] B. L. Chong García, " Desarrollo de una aplicación móvil en el ámbito deportivo", **Trabajo fin de grado,** Departamento de Informática, Universidad Carlos III de Madrid, Madrid, España, 2016. **[En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/27176.

[32] R. Cañizares Sanz, " Diseño y desarrollo de un portal eCommerce de ropa, calzado y accesorios deportivos ", **Trabajo fin de grado,** Departamento de Informática, Universidad Carlos III de Madrid, Madrid, España, 2016. **[En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/26836.

[33] R. Gutiérrez Sopedra, " Exam developer: desarrollo de una aplicación móvil enfocada a m-learning", **Trabajo fin de grado,** Departamento de Informática, Universidad Carlos III de Madrid, Madrid, España, 2016. **[En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/26834.

[34] B. Navas Torres, " Desarrollo de una plataforma social para el suministro colaborativo de piezas de repuesto", **Trabajo fin de grado,** Departamento de Informática, Universidad Carlos III de Madrid, Madrid, España, 2015. **[En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/23101.

[35] C. Muñoz Villar, " WeSweat, Geolocalización social de deportistas", **Trabajo fin de grado,** Departamento de Informática, Universidad Carlos III de Madrid, Madrid, España, 2015. **[En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/23085.

[36] I. Romera Alcalá, " Parkineo, aplicación Android para la búsqueda de parking", **Trabajo fin de grado,** Departamento de Informática, Universidad Carlos III de Madrid, Madrid, España, 2014. **[En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/22873.

[37] R. González Gröngberg, " Entorno y aplicación móvil Android "HISPANIA UC3M"", **Trabajo fin de grado,** Departamento de Informática, Universidad Carlos III de Madrid, Madrid, España, 2015. **[En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/22576.

[38] J. Arroyo Moreno, " Aplicación de comercio electrónico para pequeñas y medianas empresas a través de las tecnologías Open Source", **Trabajo fin de grado,** Departamento de

Informática, Universidad Carlos III de Madrid, Madrid, España, 2015. **[En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/22476.

[39] S. Torres Mendiola, " Desarrollo de una aplicación para la gestión de proyectos no gubernamentales", **Trabajo fin de grado,** Departamento de Informática, Universidad Carlos III de Madrid, Madrid, España, 2016. **[En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/24235.

[40] I. Vidal Hernando, " Elaboración de una aplicación social web 2.0 de notificación de mensajes entre alumnos", **Trabajo fin de grado,** Departamento de Informática, Universidad Carlos III de Madrid, Madrid, España, 2014. **[En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/22057.

[41]D. Villalba Trujillo, " Desarrollo de una plataforma backend y frontend para la gestión de contenidos", **Trabajo fin de grado,** Departamento de Informática, Universidad Carlos III de Madrid, Madrid, España, 2015. **[En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/23761.

[42] J. Lois Garrido, " Sistema backend y frontend para la gestión de nuevas ideas de negocio", **Trabajo fin de grado,** Departamento de Informática, Universidad Carlos III de Madrid, Madrid, España, 2015. **[En línea]. Disponible en:** https://e-archivo.uc3m.es/handle/10016/23722.

[43]"Microsoft Security Advisory 2862973", *Docs.microsoft.com*. **[En línea]. Disponible en:** https://docs.microsoft.com/en-us/security-updates/securityadvisories/2014/2862973. [Último acceso: 25/09/2018]

[44]"The Shappening", *Sites.google.com*. **[En línea]. Disponible en:** https://sites.google.com/site/itstheshappening. [Último acceso: 25/09/2018]

[45] Openwall, "announce - john the ripper", *openwall.com*. **[En línea]. Disponible en:** https://www.openwall.com/lists/announce/2013/05/30/1. [Último acceso: 25/09/2018]

[46] Offensive Security, *Kali.org*. [Online]. Available: https://www.kali.org/. [Último acceso: 25/09/2018]

[47]"Virtualización de VMware para escritorios, servidores, aplicaciones y clouds públicas o híbridas", *VMWare*. **[En línea]. Disponible en:** https://www.vmware.com/es.html. [Último acceso: 25/09/2018]

[48] Offensive security, *Offensive-security.com*. **[En línea]. Disponible en:** https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/. [Último acceso: 25/09/2018]

[49]"teeshop/rexgen", *GitHub*. **[En línea]. Disponible en:** https://github.com/teeshop/rexgen. [Último acceso: 25/09/2018]

[50]"magnumripper/JohnTheRipper", *GitHub*. **[En línea]. Disponible en:** https://github.com/magnumripper/JohnTheRipper. [Último acceso: 25/09/2018]

[51]"John the Ripper - cracking modes", *openwall.com*. **[En línea]. Disponible en:** https://www.openwall.com/john/doc/MODES.shtml. [Último acceso: 25/09/2018]

[52]"OWASP Rainbow Maker Project - OWASP", *Owasp.org*. **[En línea]. Disponible en:** https://www.owasp.org/index.php/OWASP_Rainbow_Maker_Project. [Último acceso: 25/09/2018]

[53]"MySQL :: MySQL 8.0 Reference Manual :: 6.2.1 Privileges Provided by MySQL", *Dev.mysql.com*. **[En línea]. Disponible en:** : https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html. [Último acceso: 25/09/2018]

[54]"Myths about /dev/urandom [2uo]", *2uo.de*. **[En línea]. Disponible en:** https://www.2uo.de/myths-about-urandom/. [Último acceso: 25/09/2018]

[55]"P-H-C/phc-winner-argon2", *GitHub*. **[En línea]. Disponible en:** https://github.com/P-H-C/phc-winner-argon2. [Último acceso: 25/09/2018]

[56]"phxql/argon2-jvm", *GitHub*. **[En línea]. Disponible en**: https://github.com/phxql/argon2-jvm. [Último acceso: 25/09/2018]

[57]"kosprov/jargon2-api", *GitHub*. **[En línea]. Disponible en:** https://github.com/kosprov/jargon2-api. [Último acceso: 25/09/2018]

[58]"Password Storage Cheat Sheet - OWASP", *Owasp.org*. **[En línea]. Disponible en:** https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet#Argon2_usage_proposal_in_Java. [Último acceso: 25/09/2018]

[59]"phxql/argon2-jvm", *GitHub*. **[En línea]. Disponible en:** https://github.com/phxql/argon2-jvm/blob/master/docs/compile-argon2.md. [Último acceso: 25/09/2018]

[60]"draft-irtf-cfrg-argon2-02 - The memory-hard Argon2 password hash and proof-of-work function", *Tools.ietf.org*. **[En línea]. Disponible en:** https://tools.ietf.org/html/draft-irtf-cfrg-argon2-02. [Último acceso: 25/09/2018]

[61]"P-H-C/phc-winner-argon2", *GitHub*. **[En línea]. Disponible en:** https://github.com/P-H-C/phc-winner-argon2/blob/master/argon2-specs.pdf. [Último acceso: 25/09/2018]

[62]"Cipher | Android Developers", *Android Developers*. **[En línea]. Disponible en:** https://developer.android.com/reference/javax/crypto/Cipher. [Último acceso: 25/09/2018]

[63]"KeyGenerator | Android Developers", *Android Developers*. **[En línea]. Disponible en:** https://developer.android.com/reference/javax/crypto/KeyGenerator. [Último acceso: 25/09/2018]

[64]"Sistema Android Keystore | Android Developers", *Android Developers*. **[En línea]. Disponible en:** https://developer.android.com/training/articles/keystore?hl=es-419. [Último acceso: 25/09/2018]

[65]"Pattern (Java Platform SE 7)", *Docs.oracle.com*. **[En línea]. Disponible en:** https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html. [Último acceso: 25/09/2018]

[66]"RFC 5321 - Simple Mail Transfer Protocol", *Tools.ietf.org*. **[En línea]. Disponible en:** https://tools.ietf.org/html/rfc5321. [Último acceso: 25/09/2018]

[67]"RFC 1034 - Domain names - concepts and facilities", *Tools.ietf.org*. **[En línea]. Disponible en:** https://tools.ietf.org/html/rfc1034. [Último acceso: 25/09/2018]

[68]"Mail::RFC822::Address", *Ex-parrot.com*. **[En línea]. Disponible en:** http://www.ex-parrot.com/~pdw/Mail-RFC822-Address.html. [Último acceso: 25/09/2018]

[69]"Input Validation Cheat Sheet - OWASP", *Owasp.org*. **[En línea]. Disponible en:** https://www.owasp.org/index.php/Input_Validation_Cheat_Sheet. [Último acceso: 25/09/2018]

[70]T. N.V., "Planificador de proyectos | Diagrama de Gantt Online | Tom's Planner", *Tomsplanner.es*. **[En línea]. Disponible en:** https://www.tomsplanner.es/. [Último acceso: 25/09/2018]

[71]"CC0 - Creative Commons", *Creative Commons*. **[En línea]. Disponible en:** https://creativecommons.org/share-your-work/public-domain/cc0/. [Último acceso: 25/09/2018]

[72]"Apache License, Version 2.0", *Apache.org*. **[En línea]. Disponible en:** https://www.apache.org/licenses/LICENSE-2.0. [Último acceso: 25/09/2018]

[73]"Various Licenses and Comments about Them- GNU Project - Free Software Foundation", *Gnu.org*. **[En línea]. Disponible en:** https://www.gnu.org/licenses/license-list.en.html#GPLCompatibleLicenses. [Último acceso: 25/09/2018]

[74] "Command and Shell User's Guide", *Roma1.infn.it*. **[En línea]. Disponible en:** http://www.roma1.infn.it/SIC/_OLD_documentazione/unix/migr/digital-unix-doc/DOCUMENTATION/HTML/AA-PS2HD-TET1_html/uc6.html. [Último acceso: 25/09/2018]

9. RESUMEN EN INGLÉS

9.1. Introduction

9.1.1. Motivation of Work

Today, computer security is gaining strength, and little by little there is a greater awareness of people for the security of their private data. Although this is true, in the field of computer engineering, security does not seem to be too well found among branch professionals, this is due to the implication of a greater complexity resulting when designing a new system since you must ensure that it is also safe.

There is a bad conception of what is computer security. Generally, when talking about computer security we think about the encryption of data with symmetric or asymmetric encryption systems. While this is true, it is not true that the solution to all security problems is to add the encryption layer to the data to avoid being read in an unauthorized manner. There is a balance that must be maintained between the security, functionality and usability of the computer system.

In some cases, computer systems require the user to be able to identify themselves in order to offer the correct functionalities. The method by which users are usually identified in front of a system are user credentials. User credentials are the means by which a user is able to identify themselves to a computer system. These are generally known as the username and password of an account. Additionally, user credentials can belong to three categories:

- **Something that the user knows.** The most common example is knowing a password, PIN, or number that is expected to be secret to the rest of the world except for the user and the system.
- **Something that the user has.** It can be a token, or a "smart card". In this case a simple example would be the DNIe that has a circuit with PKI certificates to be recognized as Spanish citizens, endorsed by the National Police (due to the PKI). Another possible example is the Passport that has personal information such as names, surnames, date of birth and keeps certificates to be authenticated with the corresponding police entities.
- **Something that the user is.** Generally, refers to biometric data that are personal data obtained from a specific technical treatment, relating to the physical, physiological or behavioral characteristics of a natural person that allow or confirm the unique identification of such person, such as facial images or fingerprint data [1].

The problem of user credentials lies in the fact that the common user, in case of having several accounts with different computer systems, usually employs similar users and passwords to avoid having to remember several user-password combinations.

It is also true that there are users who know that they should not have the same password for several platforms, but the way to choose the password is usually quite weak or simple, such as the birthday of someone you know or your own, the name of a pet, the name of a relative, etc.

Referring to the above, one of the most common attacks to obtain people's credentials is **credential stuffing**, which is generally based on using user combinations and passwords that have been obtained after an attack on a user. database or obtained in a **password dump site** to try to find a correct combination of username-password and access the user's private data.

Considering all this, it is essential to protect the confidentiality of user credentials used in computer systems.

9.1.2. Objectives

The objectives of this TFG are, firstly, to make UC3M students aware of the importance of protecting user credentials in the applications or systems they develop, no longer specifically for their TFGs, but for future systems in which they participate. Even so, the audience of this TFG is not limited to students of UC3M, but to any other computer engineering student who is not excessively related or knowledgeable about computer security practices.

In addition, several security concepts that are basic to understanding the subsequent analysis of the TFG from previous years will be explained. In this analysis, the possible vulnerabilities observed in the system / application with respect to user credentials will be briefly explained. In the event that these vulnerabilities exist and have not been addressed, solutions for them will be presented. Subsequently, statistics will be made with the data obtained from the analysis of each TFG, which will try to reflect, among others, the level of awareness regarding safety and the level of involvement, at least mentioning safety or carrying out the implementation of a security mechanism.

Finally, some didactic materials will be generated that will serve the readers as guided exercises to learn about some of the practices in the protection of user credentials previously explained.

9.2. State of Art

9.2.1. Basic security concepts

The concepts explained below are key to understand why the security mechanisms used and facilitate a better understanding of the points discussed throughout the document.

The first of the concepts is vulnerability. A vulnerability is a security breach in a computer system or a weakness in security in the system or application. It is usually due to design flaws or an implementation error that can allow an attacker to cause damage to assets or users. The assets in this area refer to the data or the application or system. As for the term users, it refers to the owners and end users who use the service offered. The latter can be end users who use the application or other entities or companies that use the service offered.

The next concept is that of threat. A threat is the intention to make an attack by an entity towards one of the vulnerabilities of the system or application. There are known threats that are usually collected in a repository, but there are also unknown threats that are those that are identified by studying the operation of the system or application for a period of time. Some of the threats related to the protection of credentials can be a Key Logger or sniffers. The key loggers can be software or hardware used to record the keystrokes and then send them to another site known by the attacker to review the data about what has been written. Sniffers on the other hand are programs that focus on the monitoring and analysis of network packets.

The last of the concepts to introduce is the attack. An attack is the action of carrying out technique to exploit the vulnerabilities of the system or application, that is, what an attacker would do to take advantage of a vulnerability. Some of the most common examples are a brute force attack, DNS poisoning, DoS attacks and DDoS (Distributed Denial of Service). The DNS

poisoning attack focuses on the introduction of corrupt data in the DNS server (Domain Name Server) that leads to the redirection of users to malicious web pages of the attacker.

9.2.2. Proposal of solutions

This section will explain what measures should be carried out to ensure that the credentials of users of the system or application being developed are secure.

Password policy

The first thing that should be done is not to limit the characters that a user can use to choose their password. While it is true that neither the character set nor the length of the password should be limited, to avoid DoS attacks it is advisable to set a limit, but it should not be too restrictive. A possible maximum for the password would be 160 characters. The fact of not limiting the character set for the password puts us at risk for XSS (Cross Site Scripting) and SQL injection attacks, therefore, it is advisable to use whitelists and blacklists to avoid the use of unwanted characters or patterns that are malicious as it can be <script> </ script>.

It is also advisable to follow a password creation policy so that the user adds a certain level of complexity to the password. One of the most common policies is that at least contain a capital letter, at least contain a lowercase letter and contain at least one digit. These guidelines increase the number of combinations necessary for brute force attacks, although it is true that they can give clues to attacks by dictionary, therefore, the user should be warned that he does not clearly modify his password to meet the minimum requirements such as You can change the first letter to uppercase and add a digit or set of digits related to your personal data at the end. What is necessary is the specification of the minimum length of the password that according to the NIST must be 8 characters, specified in section 5.1.1.1 of the digital identification guide [14].

Hash functions

Once the user is free to choose their password, it is necessary to know which methods to use in order to securely store user credentials. To do this, you should use hash functions about passwords and store the hash value obtained. In this way the user authentication can continue and in case the database where the passwords reside is exposed to an attack, the data stored on the passwords will not be immediately useful to the attackers. However, the fact of using a hash function is not enough, since, hash functions that are considered cryptographically secure and updated must be used. Some of these functions are:

- bcrypt
- scrypt
- PBKDF2
- Argon2
- SHA2
- SHA3

It is necessary to be aware that hash functions do not simply fulfill the objective of safe storage of passwords, but were designed for other purposes, that is why you should know what function is being used and if it was designed for the purpose in which it is going to use. Additionally, some of the aforementioned hash functions are designed to slow down the hashes, something that must be taken into account when it is going to be used in a system or application where the speed of response is crucial.

Cryptographic salt

Although cryptographically secure hash functions are employed, a greater degree of protection can still be added to users through the use of cryptographic salts. The reason is that several users will use similar passwords, already used by other people or easily deductible. Therefore, if the hash function is simply applied, if the attackers realize that two hashes are identical, it can only be because both users have the same password and would only have to get the password of one of the two users to access any of them. the two accounts. To avoid this type of situations, the cryptographic salts are a set of pseudo-random characters generated by CSPRNGs that are usually added to the start or end of the users' passwords before hacking the password. In this way what is hashea is a concatenation of the salt and the password, and even if two users use the same password as the salts will be different the resulting hash will also be different.

Regarding the use of the salts, one must be cautious because you should never reuse a salt value. It should not be used for more than one user. Each time the password changes or you want to have a new password, you must use a new and different salt value for each of them. When it is time to store the hash, the salt with which the hash function was used must be stored so that the authentication process can be carried out each time the user wants to log in again. The salt in this case can be stored in clear because it will be different for each user and of a length at least equal to the output size of the hash function that is used.

Data encryption

Another option is the encryption of data that is believed to be critical. The decision to use data encryption must be weighed and decide whether it is really necessary or not. In order to make a correct and argued decision one must know what each type of encryption is used for, knowing its advantages and disadvantages. The types of encryption depending on the type of key used are the following.

Symmetric encryption, which is based on the use of a secret key for the world except for the sender and receiver of the message. The message is encrypted with the secret key and sent through the channel to the receiver. The receiver, using the previously shared or agreed secret key, decrypts the message. The same key is used for encryption and decryption, so it must be secret. Some examples of symmetric encryption systems are DES, Triple DES (TDES or 3DES) and AES.

The advantages of the symmetric systems are:

- Speed, symmetric encryption is much faster than asymmetric.
- Less use of system resources compared to asymmetric encryption.
- Need for only a single key to encrypt and decrypt data.

The advantages mentioned are only those related to the use of a symmetric encryption system for data encryption of for example a database. Symmetric encryption systems have other additional advantages when they are used for the exchange of messages.

The main disadvantage in this case is that by using a single symmetric key for encryption and decryption, if it is compromised, all data would no longer be confidential.

Asymmetric encryption, also known as public key, is based on the use of two keys. A key is called a public key and as its name indicates it is public and known all over the world. The other key is

called a private key, this key must be secret and not revealed to anyone. An example of a popular asymmetric cipher is RSA.

In the case of asymmetric encryption, the way in which it is encrypted changes with respect to symmetric systems. Suppose Alicia wants to send a message to Benito, both of them have their public and private key pair. In the case that Alice wants to send an encrypted message to Benito, the message must be encrypted with Benito's public key. Once Benito receives the message, he can decrypt it if he uses his private key. Next, a scheme to clarify the operation is shown.

The main advantage of asymmetric systems is that if long keys are used the asymmetric cipher system is heavier and therefore more difficult to break. On the contrary, the disadvantages are:

- Slow compared to symmetric encryption
- Uses more system resources than symmetric encryption
- The loss of the private key results in the impossibility of deciphering the data, since public and private keys can not be derived one from the other.

The hybrid encryption is an encryption that uses the two previous types of encryption to make the most of its qualities. The hybrid encryption focuses on the use of symmetric encryption for data encryption, so that the process is fast and efficient as it ensures the confidentiality of the data. Asymmetric encryption is used in this case to protect the encryption key from symmetric encryption. In this way, asymmetric encryption is used, which is heavier, but more powerful to protect the symmetric key and the encryption of the data is done symmetrically to accelerate the process. This is a very useful method for when you want to ensure the confidentiality of the data, but a symmetric encryption falls short and an asymmetric encryption provides an excess of time and computation for the data you want to encrypt.

Additionally, it can be differentiated depending on the encryption mode used. The modes shown below are those recommended by NIST in its recommendation document for block encryption modes [15]. The recommended modes are the following:

- ECB, *Electronic Code Book Mode*
- CBC, *Cipher Block Chaining Mode*
- CFB, *Cipher Feedback Mode*
- OFB, *Output Feedback Mode*
- CTR, *Counter Mode*

There is another series of actions that can be carried out to ensure the security of user credentials without necessarily working with them. One of them is to limit the responsibilities of access to the database. This is because many people use the database administrator user more easily as a user to make queries that do not need to be done by the administrator. For these cases it is a good idea to create other users so that access to information within the database is limited.

Another action that can be carried out is the limitation of permissions to files and folders on Unix systems. In case you have a Unix system you should pay special attention to the permissions that each user has on the system. In this case it is recommended to apply the policy of prohibiting all accesses and granting only the necessary permissions against the policy of prohibiting or denying the permissions that the user should not have. The advantage of the first policy is that it ensures that the user has only the basic permissions necessary to operate in the system. The second policy is not so recommended as it depends on the ability to deny

permissions to everything that is not given access and there may be errors or that permits are not prohibited that should be denied.

Finally, it should be mentioned that although the use of HMACs is highly recommended as a solution to ensure the confidentiality of user credentials, due to the strong need to keep the key used safely and secretly so that they are a better option than Analog hash functions are not recommended for use. Since this would imply that the authors would have to think about a safe way of storing the key for the HMAC function and a secure mode of transmission of the key, which in the end would end up complicating the design of the system versus the use of one of the hash functions previously exposed.

9.3. Analysis of recent TFG

9.3.1. Analysis of TFG

In this section an individual analysis of each of the TFGs that have been taken as valid samples for the study of the protection of user credentials in the TFG of previous students of the UC3M [16] will be carried out.

The process that has been carried out has been divided into two stages, in the first stage a superficial revision of the works in which samples were filtered depending on whether it was related to the user credentials or development of an application, obviously the significant samples that are useful for this study are those that make use of users registered in the system and use an authentication system. In the second stage, once all the studied works have been filtered until the moment they are of the order of 400 records obtained from the e-File finder UC3M [17], we proceed to do a study in greater depth studying what data are stored on the users and if they use user credentials, as well as a general knowledge of the system that has been developed.

As mentioned, the e-File search engine has been used to obtain the different TFGs, for which the search parameters have been to specify the community of academic works. Afterwards, the academic works have been listed by area of knowledge, which in this case is the Informatics. Finally, they have been ordered by publication date in descending order and records are obtained that correspond to each of the TFGs stored in the UC3M e-Archive repository. Regarding the filtering of the TFG, the existence of three branches of specialization in the Degree in Computer Engineering that are Computers, Computing and Information Systems have also been taken into account. Taking into account the existence of these three branches, it is quite probable that the works related to the branch of Computers and Information Systems deal with the development of applications, as well as those of the branch of Computation are focused on the development of algorithms and techniques of Artificial Intelligence to generate bots or agents that fulfill a role in a game. Additionally, while the study was carried out, the Double Degree in Computer Engineering and Business Administration has also been taken into account, since they generally carried out the development of a system for commercial purposes, trade aid or assistance to take of decisions of other companies.

Once the end of the second stage of evaluation of the TFG was completed, a total of 25 significant samples were obtained, which at most date from four years prior to the study.

For the individual analysis of each chosen work, it has been taken into account which application has been developed and if it uses users. It should be mentioned that among the samples obtained, some have been added that do not require user registration because it is mainly

suspected that this design decision would save them time and concerns about the security that should be added to the system. The general outline that has been followed for each individual analysis is a brief introduction to the developed project in which it is explained that the system does at a general level. Subsequently, it was analyzed if the system requires users and if a registration process is carried out. In the event that the registration of personal data of the user is carried out, even if it is simply a username and password, it is checked whether throughout the document there is a section or chapter that specifies the data that is taken and if there are requirements of security that cover the confidentiality of these. Additionally, since we are dealing with personal data, in many of the cases we have taken into account the appearance of a section that refers to the legislation applicable to the system, such as the LOPD. In this case, the current RGPD has not been taken into account since it came into force at the end of 2017 and there are no TFG analyzed after that date. Finally, it is argued that the project author has taken into account the security mechanisms necessary to protect user credentials or if they have not been taken into account. In order to assign a quantitative value to this degree of security provided by the author, it has been decided to make a scale from zero to four. The levels of the security scale are explained below:

- **Grade 0:** this grade is the lowest of all and means that even though the project deals with user data and requires the use of user credentials, the author completely omits security and does not make security or reference requirements to the legislations applicable to the system.
- **Grade 1:** in this degree of security, the author makes at least mention or security requirements due to the treatment of user data or the applicable legislation is briefly mentioned. If one of the two options are given, this degree of security is obtained.
- **Grade 2:** this degree of security is the same as Grade 1, with the difference that it is necessary to mention both options in the project document.
- **Grade 3:** this degree of security extends the security grade 2, so it is expected that the form in which the security mentioned in the security requirements will be implemented will be specified or if a detailed analysis of the security has been carried out. regulatory or legal framework.
- **Grade 4:** this is the last grade and it is what one should aspire to have, in this degree mention is made of the security requirements and the decisions of the security mechanisms used are argued or the algorithm will be specified. Additionally, an-in-depth analysis of the regulatory framework or legal framework is made.

The degree of security of each of the TFGs is specified at the end of the analysis of each one of them.

9.3.2. Results

Based on the data obtained, it can be concluded that, in general, the year 2014 began with a security degree 1 that had an increase in 2015 and the beginning of 2016, and then decreased at the end of 2016 and 2017 to the previous value in 2014. all types of applications developed, it can be assured that web applications are those that have taken into account the protection of user credentials, in principle because applications used to be eCommerce portals and the data processed is critical.

Another thing that must be highlighted is the strategies that have been observed when carrying out the TFG. The strategies in question refer to how to deal with the problem of security when the project is developed. One of the strategies is to choose to develop an application or system

that does not require authentication and thus avoid the treatment of user data. Another similar strategy is based on erasing all the data generated when the system has been used, this strategy is common in simulation systems. The following is not a strategy, but a defect in how to approach the TFG. After reading the TFG it has been seen that many authors do not give the necessary importance to security because they seem to believe that being a TFG is a development of a system that will not go out into a real environment and they prefer to focus on contributing as many functionalities as possible in order to present a more attractive project. Finally, there is a group of authors who are aware that they have to develop a system with authentication and therefore treatment of user data. The problem is that in spite of knowing that they have to provide security mechanisms to protect the confidentiality of the data, they do not know how to carry them out and generally derives in requirements of the style "the database will be encrypted" and brief studies of the LOPD as "the system will be provided with the security mechanisms to comply with the LOPD".

9.4. Design of didactic materials

Based on the results obtained from the analysis of the TFG and the proposed solutions, a process of elaboration and selection of the didactic materials is carried out. With this objective in mind, it should be taken into account that of the students analyzed, the main problems that did not include mechanisms to protect user credentials are:

- They are not aware that they are sensitive data and must protect them, in part it may be because they do not know the ease with which attacks can be made.
- Or to be aware that they have to ensure the security of users' data and they only think of encrypting the database because they do not know other methods.

Another point that must be taken into account are the proposed solutions to ensure the protection of user credentials that were provided in the second chapter. The proposed solutions are the following:

- Add a password policy.
- Use hash functions.
- Add a cryptographically strong salt.
- Encryption of data arguing the reason for its use, encryption is not acceptable as default option.
- Limit the permissions and responsibilities of the entities that deal with user data.

9.4.1. Didactic Material 1

Taking into account these two sets of ideas, it was thought that one of the objectives of the didactic materials would be to attract the attention of readers so that they would realize the attacks that can be carried out and how simple they can be. With this in mind we designed the first didactic material that is expected to attract the attention of the reader, so it must be entertaining, and that some basic concepts about the treatment to be applied to passwords be presented. For the first didactic material the tool John the Ripper will be used to break hashes that are weak, in this way the reader will learn one of the tools with which they can suffer attacks and how easy it is to perform on the hash functions that are not safe. In addition, when using John the Ripper, its free version will be used to emphasize the ease of performing the attacks. Additionally, a virtual machine with Kali Linux will be used, which comes by default with a set of tools that can potentially serve to attack systems, although in the didactic material the use of these tools will not be seen.

9.4.2. Didactic Material 2

Once the reader has ideally performed the first didactic material, by having their attention captured and an increase in their level of awareness regarding the security of user credentials, we proceed to explain a method to protect user credentials that does not involve your treatment directly. To do this, an issue is introduced that will serve the reader in future design decisions in which security should be taken into account. The second didactic material will propose the limitation of the permissions and the responsibilities of those entities that treat the data of the users. For this, an approach is made to the use of MySQL databases and Unix systems. In the first case it will be proposed the creation of another user that is not the default user of MySQL to access the database, in addition you can increase the level of the limitation by creating a second user who only has access to the tables with the user data. On the other hand, the system of permissions of the files in Unix systems will be explained so that the user can generate a user and grant him only the necessary permissions for the management of the system.

9.4.3. Didactic Material 3

The third of the didactic materials has been oriented to the concept that has tried to transmit with greater impetus throughout the document, the correct use of hash functions. For this, a simple implementation of hash functions will be demonstrated first, after which the concept of cryptographic salt will be introduced to increase security against known attacks against hash functions. Finally, an implementation of the winning function of the PHC (Password Hashing Competition), which is the Argon2 function, will be introduced. In this case, the Java implementation will be carried out due to its extended use in the TFG analyzed to develop the system.

9.4.4. Didactic Material 4

In the forth of the didactic materials it is expected to provide the necessary knowledge on the types of encryption so that the students are able to discern and argue when they should use each of the types of encryption. Additionally, there is a brief review of the operation modes of block ciphers which is something that many students do not remember. For this exercise, the implementation of data encryption in Android Studio will be developed due to the fact that most of the mobile applications are developed in Android, in addition to using the classes in Java which allows its implementation also in web applications that in which It has been noticed a great use of projects implemented in Java. In the didactic material both symmetric and asymmetric encryption will be shown, but an example of hybrid encryption will not be provided, since it is trivial to do it once it is known as encrypt and decipher the data using symmetric and asymmetric ciphers.

9.4.5. Didactic Material 5

The last of the didactic materials has the objective of approaching another approach to the protection of user credentials indirectly. In this exercise we will expose the use of regular expressions to generate examples of whitelists and blacklists in Java. In this way, the user will be provided with a way to filter the data received by the system to know if they are valid or malicious.

9.4.6. Resumen de los materiales didácticos

In this section it is verified that the didactic materials proposed comply with the points exposed at the beginning of the section, for which the following table will be used.

TABLA 13: DIDACTIC MATERIALS SUMMARY

Didactic Material	Covered solutions	Motivation	Form of protection
Didactic Material 1	Password policy and hash functions.	Students do not know the need to protect user credentials	Protection of credentials directly
Didactic Material 2	Limitation of permits and responsibilities	Students do not know how to protect user credentials	Protection of credentials indirectly
Didactic Material 3	Cryptographically strong salt and hash functions	Students do not know how to protect user credentials	Protection of credentials directly
Didactic Material 4	Encryption of data and justification of the election	Students do not know how to protect user credentials	Protection of credentials directly
Didactic Material 5	Password policy	Students do not know how to protect user credentials	Protection of credentials indirectly

9.5. Conclusions

9.5.1. Conclusions and Objectives met

Once the project is finished, the objectives that have been met are analyzed and conclusions are drawn about the work carried out.

The document has been able to offer an overview of the techniques currently used for the protection of user credentials. In addition, basic security concepts have been introduced so that any reader who does not have prior knowledge of computer security is able to understand the concepts explained. Additionally, it has been possible to make a solution proposal based on current techniques, specifically those that add a minimum degree of security so that credentials are easily usurped, but without having to place an excessive burden on the following authors. TFG futures to implement them.

In addition, a study has been carried out on the degree of security with respect to user credentials in some recent TFGs. Each of these TFGs has been analyzed individually, which has allowed a greater understanding of the project in question. Based on these analyzes, an evaluation of the data obtained has been carried out, from which several hypotheses have been proposed that explain the behavior of the ups and downs of the security level over the 4 years evaluated. Even, it has exposed some of the most common strategies carried out by students when it comes to having to conceive security in their system. Because of this, it has been able to glimpse more easily the problems that students have when including the confidentiality of user credentials in their applications.

Based on the techniques studied, the proposals and the identification of the problems of the authors when faced with computer security, a series of teaching materials have been generated that manage to collect all the proposed solutions, taking into account the technologies most used by the users

9.5.2. Future lines of work

The project has several points that can be improved and re-evaluated in future work. This is due to the fact that computer security is a field that improves when attacks are made and new vulnerabilities are discovered on the methods that are currently used. Therefore, it is necessary that the project reviews the proposed algorithms with those that are in use at the time of the review to know if they are indeed providing secure algorithms or the given ones are already broken and are not cryptographically secure.

Referring to the part of the study carried out in the project, it would be interesting to investigate and analyze the security of other older TFG to see if there is a pattern or to see what events mark the differences in security. Additionally, as is logical, the study could be carried out again in four or five years to see if the project has had an impact and has helped the students to improve the degree of safety or if, on the contrary, it has not been successful. In the event that another study is carried out again, it would be interesting to increase the number of samples collected in order to have more significant statistics and draw conclusions about them. Finally, in the new study carried out it would be necessary to check the GDPR and not only take into account the LOPD regarding the laws that should affect the analyzed project.

Another possibility is that, instead of keeping the project focused on the protection of user credentials, a study could be made on other fields of computer security, such as communication protocols used and protection against known attacks.

Regarding the didactic materials it would be a great improvement to provide audiovisual material such as tutorials with an approximate duration of ten minutes in which the theoretical concepts of the exercise are explained and a guided example is carried out. In addition to these new multimedia materials, there should be generated a series of additional exercises for students, so they can practice the concepts for themselves instead of following the guided exercise.

Another possible inclusion would be to add a study of the different laws or standards that are aplicable to security and make an informative summary about them, so it contains the most important points.

ANEXO. DATOS DE LOS TFG ANALIZADOS

Número de TFG	Título del Trabajo Fin de Grado
1	Herramienta de análisis de legibilidad de contenidos educativos
2	Modelado de un puerto marítimo con Unity 3D
3	Desarrollo de una aplicación multimodal para apoyar el estudio mediante m-learning
4	Aplicación nutricional y seguimiento deportivo, en atletas de alto rendimiento
5	Aplicación móvil para la detección de pasos basada en acelerometría a baja velocidad
6	Diseño e Implementación de un Sistema de Control Remoto para Vigilancia en Android
7	Desarrollo de gestor de comidas a domicilio mediante la aplicación de mensajería instantánea Telegram
8	Desarrollo de un asistente multimodal educativo para dispositivos móviles Android
9	Diseño e implementación de un sistema domótico basado en Raspberry Pi
10	Simulación de trayectorias de barcos y aplicación al control marítimo
11	Estrategias de diversificación eficiente de carteras e implementación de una plataforma digital de inversión
12	Desarrollo de una aplicación de turismo gastronómico para dispositivos iOS y análisis estadístico
13	Diseño e implementación de Campus Life, una aplicación móvil para la consulta e inscripción de actividades deportivas y culturales
14	Desarrollo de una aplicación móvil en el ámbito deportivo
15	Diseño y desarrollo de un portal eCommerce de ropa, calzado y accesorios deportivos
16	Exam developer: desarrollo de una aplicación móvil enfocada a m-learning
17	Desarrollo de una plataforma social para el suministro colaborativo de piezas de repuesto
18	WeSweat, Geolocalización social de deportistas
19	Parkineo, aplicación Android para la búsqueda de parking
20	Entorno y aplicación móvil Android "HISPANIA UC3M"
21	Aplicación de comercio electrónico para pequeñas y medianas empresas a través de las tecnologías Open Source
22	Desarrollo de una aplicación para la gestión de proyectos no gubernamentales
23	Elaboración de una aplicación social web 2.0 de notificación de mensajes entre alumnos
24	Desarrollo de una plataforma backend y frontend para la gestión de contenidos
25	Sistema backend y frontend para la gestión de nuevas ideas de negocio

Número de TFG	Autor/a	Fecha de edición	Fecha de defensa	Grado de seguridad	Tipo de aplicación
1	Mata San Juan, Henar	27/02/2017	01/09/2016	1	W
2	Bernández Martínez, Alejandro	27/02/2017	01/09/2016	2	P
3	Villegas Flores, Cristhian Edmundo	31/03/2017	14/03/2017	1	M
4	Vasldés Fernández, José	27/02/2017	01/09/2016	1	W
5	Labrado Vicente, Víctor	30/06/2017	05/07/2017	2	M
6	Ramírez Marcos, Víctor Manuel	30/06/2017	07/07/2017	1	M
7	Rivero Ortiz, Antonio	23/02/2017	01/09/2016	2	P
8	Muñoz Moreno, Adrián	23/02/2017	26/08/2016	0	M
9	Santos Senra, Héctor	01/02/2017	10/03/2017	0	P
10	Vázquez Coll, Jaime	10/10/2016	01/06/2016	1	P
11	Pérez Moscoso, Carlos Eduardo	06/10/2016	01/02/2015	2	W
12	San José de Vicente, Martín	03/10/2016	01/01/2016	2	M
13	Sánchez Muñoz, Daniel	01/10/2016	04/10/2016	0	M
14	Chong García, Benjamin Lee	01/09/2016	03/10/2016	0	M
15	Cañizares Sanz, Roberto	01/07/2016	13/07/2016	1	W
16	Gutiérrez Sopedra, Roberto	01/06/2016	07/07/2016	0	M
17	Navas Torres, Borja	01/06/2015	01/06/2015	3	M
18	Muñoz Villar, Carlos	24/06/2015	01/06/2015	1	M
19	Romera Alcalá, Iván	01/06/2014	01/06/2014	1	M
20	González Grönberg, Rafael	30/07/2015	30/07/2015	1	M
21	Arroyo Moreno, Javier	07/03/2016	01/10/2015	2	W
22	Torres Mendiola, Sofía	08/03/2016	08/03/2016	2	W
23	Vidal Hernando, Ignacio	01/12/2015	01/05/2014	1	W
24	Villalba Trujillo, Daniel	14/10/2015	14/10/2015	1	W
25	Lois Garrido, Javier	06/10/2015	06/10/2015	2	W